CURSO PRÁTICO 42 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 27,00

INPUT

Vol. 3 N.º 42

NESTE NÚMERO

PROGRAMAÇÃO DE JOGOS

MÓDULO LUNAR: COMANDE O POUSO

Gráficos de alta resolução. Paisagem lunar. Marcador de combustível. Velocímetro. Controles de pouso. Personalização do programa. Efeitos sonoros. Colisões. O módulo 821

CÓDIGO DE MÁQUINA

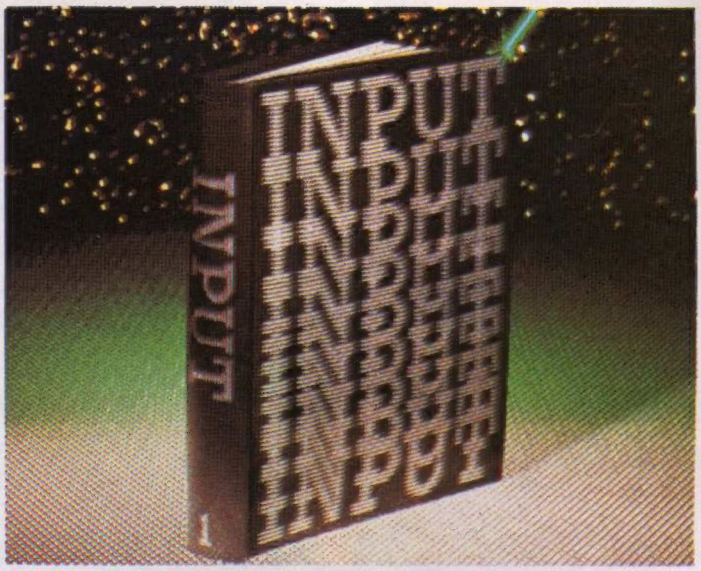
AVALANCHE: MONTE O CENÁRIO

Rotina Assembly responsável pela criação e pelo **SCROLL** horizontal no Spectrum. MSX e TRS-Color: modo texto. Uso da tabela criada pelo programa BASIC para definir o contorno da montanha. A montagem do cenário. O uso dos blocos gráficos. Como colorir os espaços. O placar. Redefinição de caracteres 824

APLICAÇÕES

UMA AGENDA ELETRÔNICA (1)

Calendário mensal. Cálculo de um evento. Calendário anual. Agenda diária. Registro de eventos. Finanças, encontros, celebrações e feriados. Como usar a agenda. Primeira parte do programa: rotinas e tela do menu principal 834



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o n.º (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

MÓDULO LUNAR: COMANDE O POUSO

- PAISAGEM LUNAR
- MARCADOR DE COMBUSTÍVEL
- VELOCÍMETRO
- CONTROLES DE POUSO

Você vai precisar de muita habilidade e sangue-frio para fazer uma nave pousar na superfície lunar em segurança. O jogo é envolvente e facílimo de programar. Confira!

Programação de jogos nem sempre é sinônimo de longas e complicadas listagens. Como você verá neste artigo, podemos criar jogos muito interessantes com programas relativamente simples e curtos.

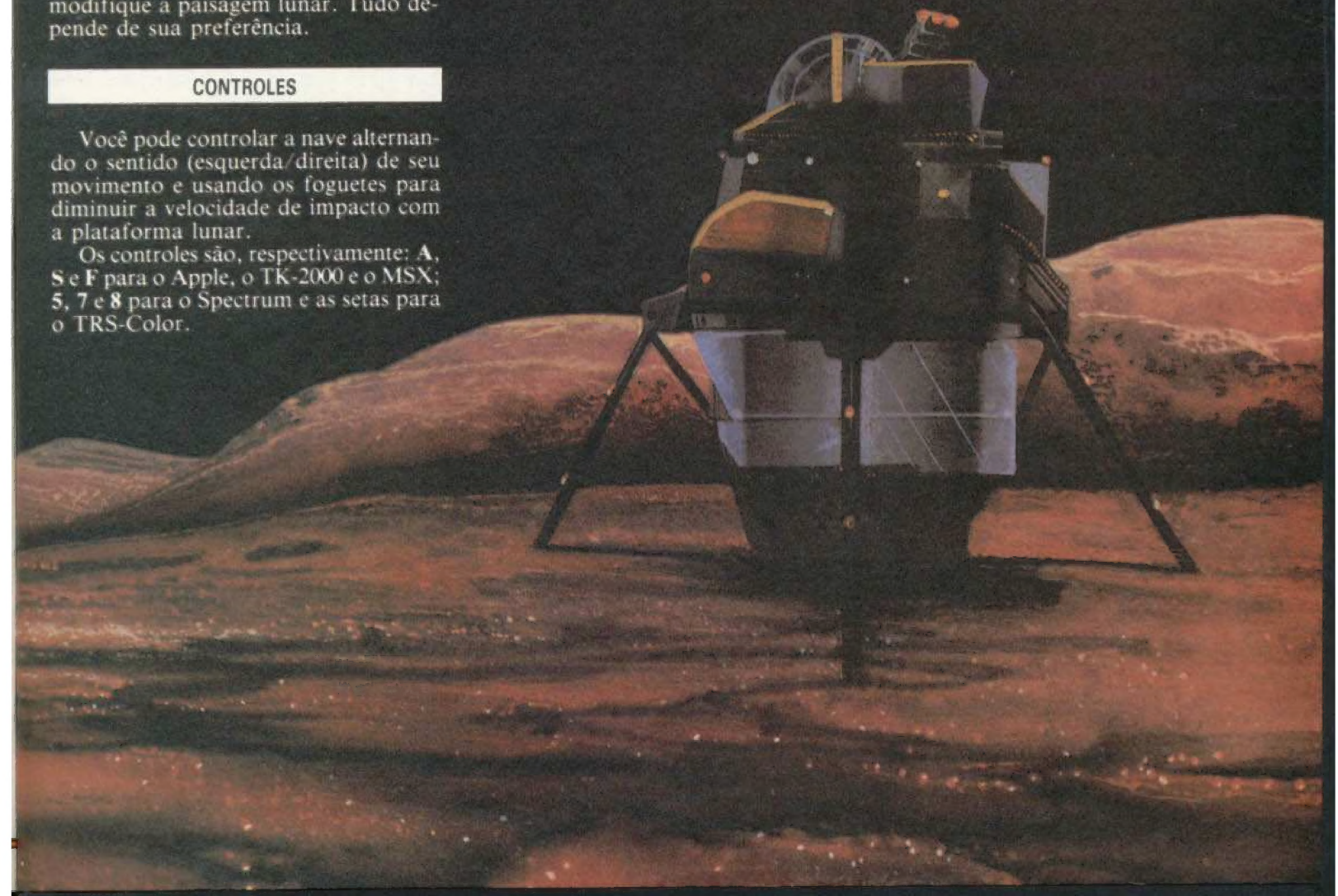
Nosso jogo, Módulo Lunar, está completo. Ele oferece ao jogador gráficos de alta resolução e controle total sobre o módulo. Se você quiser, acrescente-lhe sons, novas mensagens — como "Quer jogar novamente?" — ou modifique a paisagem lunar. Tudo depende de sua preferência.

CONTROLES

Você pode controlar a nave alternando o sentido (esquerda/direita) de seu movimento e usando os foguetes para diminuir a velocidade de impacto com a plataforma lunar.

Os controles são, respectivamente: **A**, **S** e **F** para o Apple, o TK-2000 e o MSX; **5**, **7** e **8** para o Spectrum e as setas para o TRS-Color.

```
10 COLOR 15,1,1:SCREEN2
20 FORN=1 TO 50:PSET(RND(1)*255
,RND(1)*140),15:NEXTN
30 DRAW"C6BM0,188M+18,-30M+18,+
15M+18,+8M+18,-8M+16,-20M+16,-5
M+13,+20M+14,+8M+18,+4M+10,0M+1
0,-10M+20,-25M+10,+20M+10,+10M+
20,+5M+18,-20M255,191"
40 PAINT(10,191),6
50 DRAW"C1BM145,188M+2,-4M+14,0
M+2,+4"
80 DRAW"BM93,157C1D31R1U31"
90 S=-TIME:LX=INT(RND(S)*240)+1
0:LY=INT(RND(S)*10)+15:XV=INT(R
ND(S)*10):YV=0:F=256
100 AX=LX:AY=LY
110 GOSUB 4000
120 GOSUB 1000:GOSUB 2000:GOSUB
 3000
130 IF LY<173 THEN 120
140 FOR TT=0 TO 200:NEXT
150 CLS:SCREEN0:COLOR 1,15,15
160 IF LX<147 OR LX>161 THEN 19
0
170 IF ABS(YV)>4 THEN 200
180 LOCATE5,10:PRINT"parabéns,
pouso bem sucedido":GOTO 210
190 LOCATE7,10:PRINT"!! fora da
 plataforma !!":GOTO 210
200 LOCATE0,10:PRINT"nave danif
icada, velocidade inadequada"
210 END
1000 IF LY>1 THEN GOSUB 4000
1010 LX=LX+XV:LY=LY+YV
1020 IF LX<5 THEN LX=LX+245
1030 IF LX>250 THEN LX=LX-242
1040 IF LY<1 THEN RETURN
1050 IF LY>150 THEN RETURN
1060 GOSUB 4000
1070 RETURN
2000 YV=YV+.5:IF F<1 THEN RETUR
N
2010 I$=INKEY$:IFI$=""THEN2010
```



```
2020 IFIS="F" AND F>3 THEN YV=Y
V-1:F=F-3:RETURN
2030 IFIS="A" THEN XV=XV-.5:F=F
-1:RETURN
2040 IFIS="S" THEN XV=XV+.5:F=F
-1:RETURN
2050 RETURN
3000 GS=STR$(INT((256-F)/8))
3010 DRAW"C6BM93,157D"+GS+"R1U"
+GS
3020 DRAW"C6BM105,155D35"
3030 DRAW"C15BM105,173M+0,"+STR
$(INT(YV))
3040 RETURN
4000 DRAW"C1BM"+STR$(AX)+","+ST
R$(AY)+"M-5,+10M+5,-2M+5,+2M-4,
-10"
4010 DRAW"C15BM"+STR$(INT(LX))+
","+STR$(INT(LY))+"M-5,+10M+5,-
2M+5,+2M-4,-10"
4020 AX=INT(LX):AY=INT(LY):RETU
RN
```

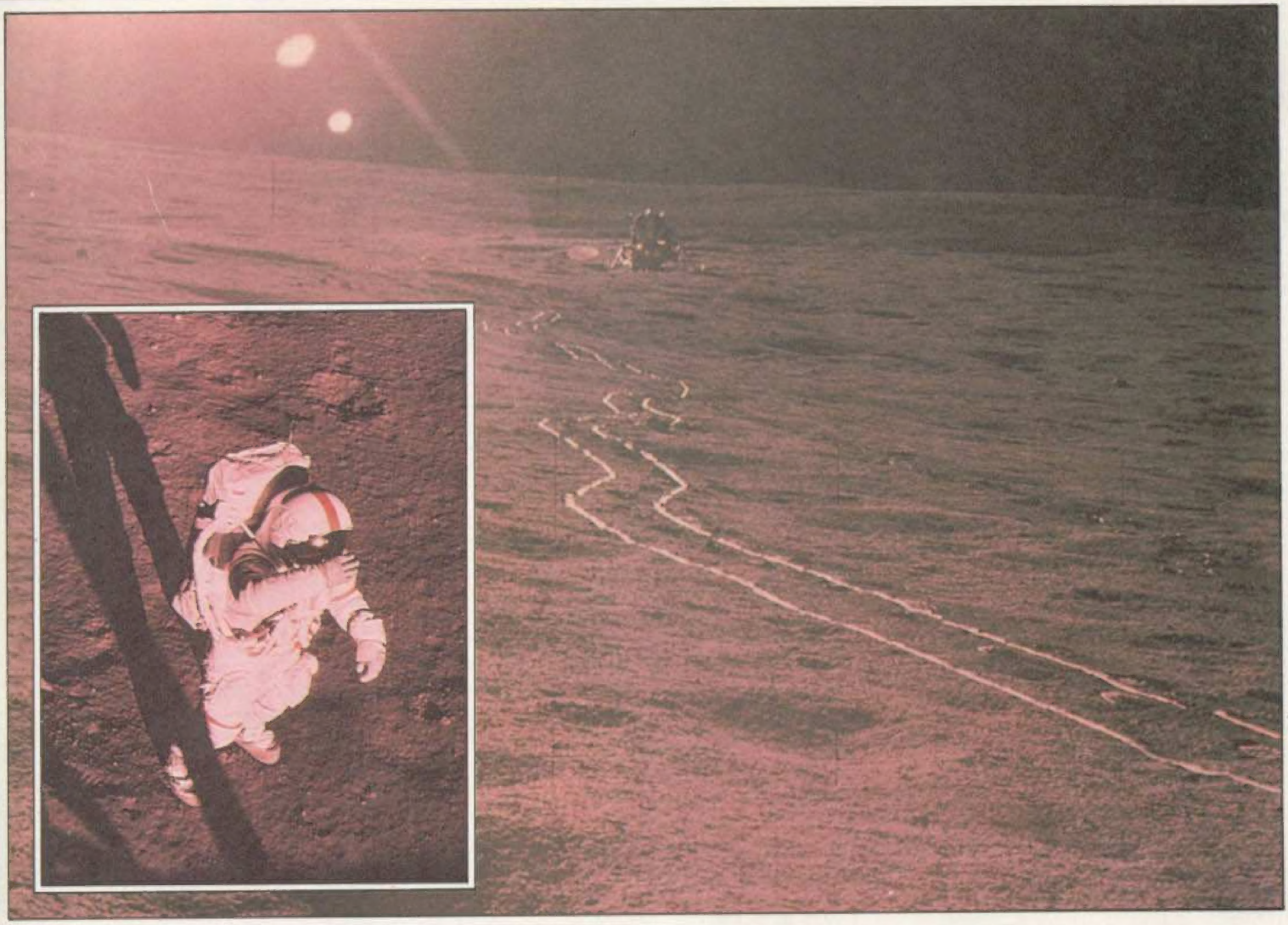
```
10  HOME : HGR : HCOLOR= 3
20  FOR N = 1 TO 50: HPLOT  RND
 (1) * 280, RND (1) * 120: NEXT
 N
70  HPLOT 0,160: FOR N = 1 TO 1
7: READ GX,GY: HPLOT  TO GX,GY:
 NEXT N
75  HPLOT 152,160 TO 152,155: H
PLOT  TO 166,155: HPLOT  TO 166
,160
80  DATA    18,130,36,145,54,15
3,72,145,88,125,104,120,117,140
,133,148,151,152,166,152,176,14
2,196,117,206,137,216,147,236,1
52,254,132,279,159
90 LX =  INT ( RND (1) * 240) +
 10:LY =  INT ( RND (1) * 10) +
 15:XV =  INT ( RND (1) * 15):Y
V = 0:F = 246
100 AX = LX:AY = LY
115  GOSUB 4000
120  GOSUB 1000: GOSUB 2000: GO
SUB 3000
130  IF LY < 140 THEN  GOTO 120

135  FOR TT = 0 TO 100: NEXT
140  HOME : IF LX < 152 OR LX >
 166 OR  ABS (YV) > 4 THEN  GOT
O 160
150  VTAB (24): PRINT "PARABENS
, POUSO BEM SUCEDIDO": GOTO 170

160  VTAB (24): PRINT  TAB( 10)
;"!!! VOCE COLIDIU !!!"
170  FOR TT = 0 TO 2500: NEXT
180  STOP
1000  IF LY > 1 THEN  GOSUB 400
0
1010 LX = LX + XV:LY = LY + YV
1030  IF LX < 5 THEN LX = LX +
245
1035  IF LX > 250 THEN LX = LX
- 242
1036  IF LY < 1 THEN  RETURN
1037  IF LY > 150 THEN  RETURN

1040  GOSUB 4000
1050  RETURN
2000 YV = YV + .5: IF F < 1 THE
N  RETURN
2010  GET IS: IF IS = "F" AND F
 > 3 THEN YV = YV - 1:F = F - 3
: RETURN
2020  IF IS = "A" THEN XV = XV
- .5:F = F - 1: RETURN
2030  IF IS = "S" THEN XV = XV
+ .5:F = F - 1: RETURN
2040  RETURN
3000  HOME : VTAB (23): PRINT "
COMBUSTIVEL: ";F
3010  PRINT : PRINT "VELOCIDADE
: "; INT (YV)
3020  VTAB (1): RETURN
4000  HCOLOR= 0
4010  HPLOT AX,AY: HPLOT  TO AX
 - 5,AY + 10: HPLOT  TO AX,AY +
 8: HPLOT  TO AX + 5,AY + 10: H
PLOT  TO AX + 1,AY
4015  HCOLOR= 3
4020  HPLOT LX,LY: HPLOT  TO LX
 - 5,LY + 10: HPLOT  TO LX,LY +
 8: HPLOT  TO LX + 5,LY + 10: H
PLOT  TO LX + 1,LY
4025 AX = LX:AY = LY: RETURN
```

```
10 PMODE 4,1
20 SS=PEEK(186)*256:DIML(1),B(1
)
30 FOR K=0 TO 7:READ A:POKE K*3
2+SS,A:NEXT
40 GET(0,0)-(7,7),L,G
50 DATA 24,60,90,126,126,90,129
,129
60 PCLS:SCREEN 1,1
70 DRAW"BM0,191":FOR X=0 TO 256
 STEP 16:READ A:LINE-(X,A),PSET
:NEXT:PAINT(127,191)
80 DATA 151,173,177,165,146,120
,167,174,177,181,181,170,140,12
2,158,170,161
90 DRAW"BM1,1S8NRDNRDBDNRDRDL"
100 LINE(8,1)-(254,5),PSET,BF:L
INE(132,7)-(132,12),PSET
110 LX=RND(248)-1:LY=15+RND(10)
:XV=RND(15)-8:YV=0:F=246
120 GOSUB 1000:GOSUB 2000:GOSUB
 3000
130 IF LY<174 THEN 120
140 CLS:IF LX<144 OR LX>153 OR
YV>4 THEN 160
150 PLAY"T1004AGFEGFE":PRINT @2
25,"PARABENS, POUSO BEM SUCEDID
O!":GOTO 170
160 PLAY"T1000ZADEFGBCDEFA":PUT
(LX,LY)-(LX+6,LY+7),L,PSET
170 FOR G=1 TO 4000:NEXT
180 END
1000 IF LY>12 THEN PUT(LX,LY)-(
LX+7,LY+7),B,PSET
1010 LX=LX+XV:LY=LY+YV:S=255-(2
55 AND LY):SOUNDS-(S-0),1
1020 IF LY<13 THEN RETURN
1030 IF LX<0 OR LX>247 THEN LX=
```



```
-247*(LX>0)
1040 GET(LX,LY)-(LX+7,LY+7),B,G
1050 PUT(LX,LY)-(LX+7,LY+7),L,P
SET:RETURN
2000 YV=YV+.5:IF F<1 THEN RETUR
N
2010 IF PEEK(341)=247 AND F>3 T
HEN YV=YV-1:F=F-3:RETURN
2020 IF PEEK(343)=247 THEN XV=X
V-.5:F=F-1:RETURN
2030 IF PEEK(344)=247 THEN XV=X
V+.5:F=F-1
2040 RETURN
3000 LINE (9+F,1)-(12+F,5),PRES
ET,BF
3010 V=2*YV:IF ABS(V)>122 THEN
V=122*SGN(V)
3020 LINE(8,8)-(255,11),PRESET,
BF:LINE(132,8)-(132+V,11),PSET,
BF
3030 RETURN
```

5

```
10 BORDER 1: INK 7: PAPER 0:
 CLS : BRIGHT 1
20 FOR N=1 TO 50: PLOT RND*
255,(RND*135)+40: NEXT N
70 PLOT 0,0: FOR N=1 TO 16:
 READ GX,GY: DRAW GX,GY: NEXT
 N
 80 DATA 18,30,18,-15,18,-8,18
 ,8,16,20,16,5,13,-20,16,-8,18
 ,-4,15,0,10,10,20,25,10,-20,
 10,-10,20,-5,18,20
 90 PRINT AT 0,4; INK 6; PAPER
 2;"COMB:";AT 0,18;"VELOCID :"
110 LET LX=RND*240+10: LET LY=
160-(15+(RND*10)): LET XV=RND*
15: LET YV=0: LET F=246
115 GOSUB 4000
120 GOSUB 1000: GOSUB 2000:
GOSUB 3000
130 IF LY>20 THEN  GOTO 120
135 PAUSE 50
140 CLS : IF LX<154 OR LX>164
OR ABS YV>4 THEN  GOTO 160
150 PRINT " PARABENS , POUSO B
EM SUCEDIDO!": RESTORE 5000:
FOR N=1 TO 14: READ A,B: SOUND
 A,B: NEXT N: GOTO 170
160 PRINT AT 10,7; FLASH 1;
INK 2; PAPER 7;"!!!!!CRASH!!!!
!": FOR T=1 TO 50: BORDER RND*
7: SOUND .01,RND*5: NEXT T
170 PAUSE 400
 180 STOP
1000 IF LY<160 THEN  GOSUB 4000
1010 LET LX=LX+XV: LET LY=LY+YV
: IF LY<300 THEN  SOUND .02,LY/
5
1030 IF LX<5 THEN  LET LX=LX+24
5
1035 IF LX>250 THEN  LET LX=LX-
242
1036 IF LY>160 THEN  RETURN
1037 IF LY<10 THEN  RETURN
1040 GOSUB 4000
1050 RETURN
2000 LET YV=YV-.5: IF F<1 THEN
 RETURN
2010 IF INKEY$="7" AND F>3 THEN
  LET YV=YV+1: LET F=F-3: RETUR
N
2020 IF INKEY$="5" THEN  LET XV
=XV-.5: LET F=F-1: RETURN
2030 IF INKEY$="8" THEN  LET XV
=XV+.5: LET F=F-1: RETURN
2040 RETURN
3000 PRINT AT 0,10;" "+STR$ F+"
 ";AT 0,28;" "+STR$ INT YV+" "
3010 RETURN
4000 OVER 1: PLOT LX,LY: DRAW -
5,-10: DRAW 5,2: DRAW 5,-2: DRA
W -4,10
4010 OVER 0: RETURN
5000 DATA .2,4,.2,7,.2,5,.2,12,
.2,0,.2,4,.2,4,.2,5,.6,7,.2,12,
.2,0,.2,4,.2,2,.6,0
```

AVALANCHE: MONTE O CENÁRIO

Não há avalanche sem montanha. Chegou a hora de colocar em cena a encosta que Willie irá escalar. Precisaremos também colorir a superfície e completar a tela com um belo céu azul.

O título e os créditos foram exibidos. O jogador já leu as instruções e ouviu a execução do tema de abertura. Chegou a hora de abrir as cortinas — ou, no caso de *Avalanche*, a hora de desenhar o cenário. No Spectrum, este se movimenta da esquerda para a direita, enquanto a tela de instruções é arrastada em sentido oposto — ou seja, realiza-se um **SCROLL** horizontal. No MSX e no TRS-Color, a tela de instruções está no modo texto. Por isso, enquanto o cenário é colocado na tela gráfica, a página de instruções vai se apagando, cedendo-lhe lugar. O deslocamento do cenário para o vídeo se dá da mesma maneira.

O processo de montagem do cenário é bem simples. Um programa BASIC cria uma tabela com o contorno da montanha (para o TRS-Color isso foi dado no artigo anterior). A partir do conteúdo da tabela, a rotina em código desenha o perfil da encosta. O céu e a terra podem ser coloridos por meio de blocos gráficos, colocados acima e abaixo do perfil da encosta. À medida que o cenário é transferido para a tela, vamos retirando a página de instruções.

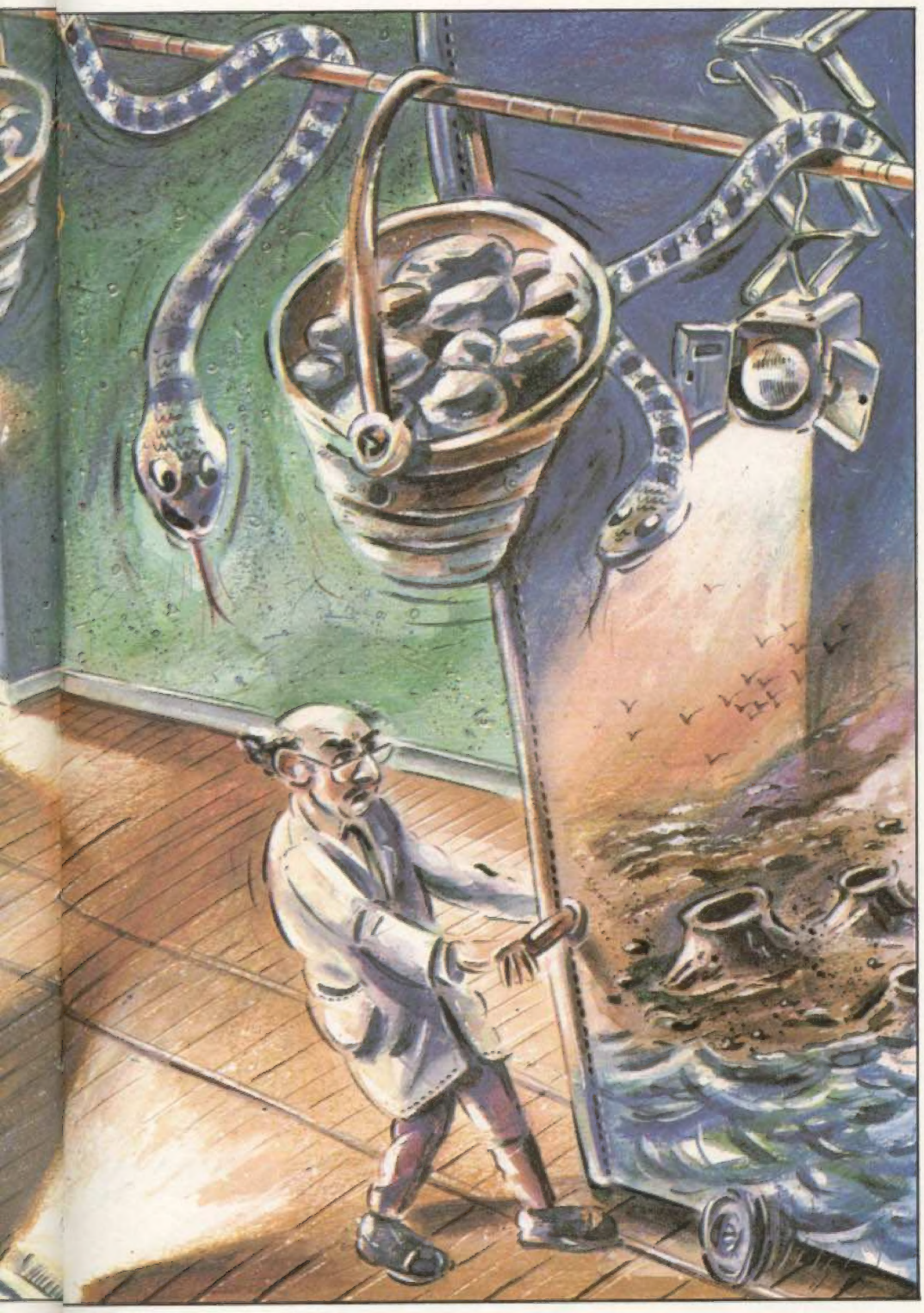
S

A rotina Assembly listada abaixo é responsável pela criação e pelo **SCROLL** horizontal do cenário.

```
10 REM org 58303
20 REM lsi ld a,16
30 REM ld (57328),a
40 REM ld ix,58034
50 REM ld b,32
60 REM mpi push bc
70 REM call scl
80 REM ld a,0
90 REM ld (57329),a
100 REM ld a,(ix+0)
110 REM dec ix
120 REM cp 33
130 REM jr nz,lv
140 REM dec b
150 REM ld a,(57328)
160 REM dec a
170 REM ld (57328),a
180 REM ld a,1
190 REM ld (57329),a
200 REM lv ld a,(57328)
210 REM ld b,a
220 REM ld hl,31
230 REM ld a,45
240 REM call lg
250 REM ld bc,57264
260 REM ld a,(57329)
270 REM cp 1
280 REM jr nz,mp
290 REM ld bc,57272
300 REM mp ld a,44
310 REM call print
320 REM ld a,(57328)
330 REM ld b,a
340 REM ld a,23
350 REM sub b
360 REM ld b,a
370 REM ld a,32
380 REM ld de,32
390 REM add hl,de
400 REM call lg
410 REM pop bc
420 REM djnz mpi
430 REM ld hl,49
440 REM ld b,12
450 REM ld a,41
460 REM ld ix,57973
470 REM call me
480 REM ld hl,113
490 REM ld b,7
500 REM call me
510 REM call elb
520 REM ret
530 REM scl ld hl,16384
540 REM ld b,216
550 REM lpi ld c,31
560 REM lpj inc hl
570 REM ld a,(hl)
580 REM dec hl
590 REM ld(hl),a
600 REM inc hl
610 REM dec c
620 REM jr nz,lpj
630 REM inc hl
640 REM djnz lpi
650 REM ret
660 REM lg push bc
670 REM ld bc,15616
680 REM call print
690 REM ld de,32
700 REM add hl,de
710 REM pop bc
720 REM djnz lg
730 REM ret
740 REM elb ret
750 REM org 58146
760 REM me *
770 REM org 58217
780 REM print *
```

Este programa BASIC coloca na memória do microcomputador a tabela cujo conteúdo é responsável pela definição do perfil da encosta.

- COMO USAR UMA TABELA PARA DEFINIR O CONTORNO DA MONTANHA
- DESLOCAMENTO DO CENÁRIO
- O USO DOS BLOCOS GRÁFICOS
- COMO COLORIR OS ESPAÇOS
- REDEFINIÇÃO DE CARACTERES



```
 5 CLEAR 57000
10 FOR n=57973 TO 58034
20 READ a: POKE n,a: PRINT n;
" ";CHR$ a
30 NEXT n
40 DATA 83,67,79,82,69,45,48,
48,48,48,48,76,73,86,69,83,45
,53,71,65,77,69,32,79,86,69,
82,32,33,33,33,35,35,33,35,35
,35,33,35,35,33,35,33,35,35,
35,35,33,35,33,35,35,35,35,33
,35,35,33,35,35,35,54
```

Como de costume, o programa usa um laço **FOR...NEXT** para colocar dados na memória. O endereço inicial da tabela criada está na linha 10. A linha 5 protege o topo da memória para que os dados das outras tabelas não sejam apagados. Na linha **DATA** os valores 33 correspondem a porções inclinadas do perfil da encosta, e os números 35, a porções planas. Os demais valores são códigos das letras utilizadas para escrever o *score* e o número de vidas que restam a Willie.

A MONTAGEM DO CENÁRIO

Após a definição do endereço inicial, temos duas instruções cuja função é colocar no endereço 57328 o valor 16. Esse número corresponde à coordenada Y do extremo superior direito do horizonte. A posição 57328 será usada para armazenar números, como se fosse uma variável.

Em seguida, a instrução **ld ix,58034** coloca no par IX o último byte da tabela que define o perfil da encosta. Esse byte define o declive do extremo superior direito da montanha.

O registro B será usado como contador do número de colunas já desenhadas. Por isso, a instrução **ld b,32** coloca nele o valor 32, correspondente ao número de colunas da tela. A próxima linha guarda esse valor na pilha.

Em seguida, a sub-rotina **scl** é chamada. Ela se encarrega da translação do conteúdo da tela uma coluna para a esquerda — ou seja, realiza um **SCROLL** horizontal.

Uma segunda posição da memória, com endereço 57329, será usada como indicador. Ela informará à rotina se o

nível da encosta está diminuindo ou permanece plano naquela coluna. Se o byte for 0, a encosta é plana; se for 1, o nível está diminuindo. As instruções das linhas 80 e 90 colocam no endereço o valor inicial 0.

A instrução **ld a,(ix+0)** coloca no acumulador o último byte da tabela de contorno. Temos que usar o "+0" nessa instrução porque o endereçamento indireto com o registro IX deve ser indexado. Não existe uma instrução **ld a,(ix)**. A instrução **dec ix** subtrai uma unidade do conteúdo de IX, para que ele aponte para o próximo valor da tabela. Observe que esta será lida "de trás para a frente".

A instrução **cp 33** tem a função de comparar o conteúdo do acumulador com 33, valor indicativo de que o nível da encosta está diminuindo. Se o acumulador não contém esse valor — ou seja, se a montanha é plana no local em questão —, a instrução **jr nz,lv** salta em direção ao rótulo **lv**.

Se o conteúdo do acumulador for 33, a instrução **jr nz,lv** é ignorada, pois o indicador de zeros foi ativado por **cp 33**. O contador B e a coordenada Y do perfil (armazenada em 57328) são diminuídos em uma unidade. O indicador armazenado no endereço 57329 passa a valer 1. Todo esse processo é executado pelas linhas 140 a 190.

A instrução **dec b** diminui o valor de B. As alterações do conteúdo dos endereços 57328 e 57329 são feitas com auxílio do acumulador, já que não existe uma instrução que, diretamente, diminua o valor de um byte da memória ou transfira um valor para determinado endereço. Não há instruções do tipo **dec 57328** ou **ld (57329),1**.

Quer a encosta da montanha apresente inclinação, quer continue plana, o programa continua na rotina **lv**.

A ROTINA lv

A linha 200 e a linha 210 colocam o valor da coordenada Y no registro B. Isso é feito com o auxílio do acumulador, já que o registro B não recebe o valor de um byte da memória via endereçamento indireto. Não há uma instrução **ld b,(57328)**.

A instrução **ld hl,31** coloca em HL a posição da memória de vídeo que corresponde ao canto superior direito da tela. **ld a,45** coloca no acumulador o valor 45, que, quando transferido para a tabela de atributos, resultará num caractere desenhado em ciano sobre fundo ciano. A rotina **lg** é chamada a seguir. Ela desenha uma coluna de blocos a partir da posição determinada por HL. A cor dos blocos é definida por A, enquanto o tamanho da coluna é estabelecido por B.

Quando o processador retorna da sub-rotina, a instrução **ld bc,57264** coloca em BC o endereço inicial do padrão do bloco usado para desenhar porções planas da encosta. O indicador guardado no endereço 57329 é colocado no acumulador e comparado ao número 1.

Se o valor do indicador for diferente de 1 — ou seja, se a montanha for plana neste local —, a instrução **jr nz,mp** faz com que o processador salte em direção ao rótulo **mp**. Se o valor do indicador for 1, o nível da encosta deve diminuir. A instrução **ld bc,(57329)** modifica o conteúdo de BC, de modo que esse par de registros passe a conter o endereço inicial do padrão do bloco usado para desenhar porções inclinadas da montanha. Os padrões dos blocos das porções planas e inclinadas da montanha são obtidos na tabela de padrões criada pelo programa listado no último artigo.

Quer a montanha seja plana ou inclinada neste ponto do desenho, o programa prossegue na rotina **mp**.

A ROTINA mp

Para desenhar os blocos que definem o perfil da encosta da montanha, determinando o limite entre o céu e a terra, o computador precisa de duas cores. Assim, a instrução **ld,a,44** atribui aos caracteres a cor verde. A cor do fundo permanece ciano. Em seguida, a sub-rotina **print**, listada no primeiro artigo da série *Avalanche*, é chamada para desenhar o bloco.

As instruções contidas nas linhas 320 e 330 colocam o valor da coordenada Y do perfil da encosta em B, com o auxílio do acumulador. Subtraindo esse valor de 23, obtém-se o número de blocos que devem ser desenhados abaixo do horizonte. O número 23, correspondente às linhas da tela do Spectrum, foi colocado em A para permitir a subtração realizada pelo comando **sub b**. Esse comando deixa o resultado da subtração no registro A. A instrução **ld b,a** é utilizada para trazer o resultado da subtração de volta para B — a rotina **lg** exige que o número de blocos a serem desenhados esteja em B. Note que todas as 24 linhas da tela do Spectrum são empregadas, incluindo as duas inferiores, normalmente reservadas para edição de linhas em BASIC.

A instrução **ld a,32** coloca em A o código de atributo correspondente a caractere de cor verde sobre fundo igualmente verde. Em seguida, o número 32 é colocado em DE e somado ao endereço de impressão na tela, que está em HL. Esse par de registros passa, então, a apontar para a próxima posição na vertical. A rotina **lg** é chamada a seguir para imprimir os blocos verdes abaixo do horizonte.

A instrução **pop bc** recupera da pilha o contador de colunas. A instrução **djnz** diminui seu valor em uma unidade, retornando ao início do programa para imprimir uma nova coluna, enquanto o contador não for zero.

O PLACAR

Precisamos reservar uma porção da tela para imprimir o *score* e o número de vidas que restam a Willie. Assim, na linha 430, 49 é colocado em HL, determinando a posição de impressão do placar. O número de caracteres impressos é colocado em B. A instrução **ld a,41** coloca em A o código de atributo correspondente a caractere azul sobre fundo ciano.

Depois, a instrução **ld ix,57973** coloca em IX o endereço do byte que será utilizado para armazenar o total de pontos obtidos pelo jogador. A rotina **me**, criada no primeiro artigo desta série, é, então, chamada. Ela traduz o *score* sob a forma de códigos ASCII, e o imprime na tela.

A instrução seguinte chama a sub-rotina **elb**, que cuida dos níveis de dificuldade do jogo, acrescentando os buracos e as cobras ao cenário. Essa rotina ainda não foi publicada e, como você pode observar, seu rótulo corresponde a um simples **ret**, no final da listagem. Por enquanto, o processador retornará imediatamente da rotina, sem nada executar. No momento apropriado, o comando **ret** será apagado pela verdadeira rotina **elb**.

Ao retornar da sub-rotina **elb**, o processador encontra uma nova instrução **ret**, que provoca um retorno ao interpretador BASIC. Quando o videogame estiver completo, a rotina de criação do cenário terá terminado e o processador retornará à rotina principal (responsável pelo controle do jogo), que chamará as sub-rotinas seguintes.

SCROLL HORIZONTAL

O rótulo **scl** marca o início da rotina que executa um deslocamento horizontal do conteúdo da tela para a direita. Esse processo — chamado geralmente

de **SCROLL** horizontal — permite que desenhemos o cenário, a partir da extremidade esquerda do vídeo, enquanto a página de instruções vai sendo retirada da tela.

A instrução **ld hl,16384** coloca o endereço inicial da memória de vídeo em HL. A instrução seguinte coloca em B o número de linhas da memória de vídeo e da tabela de atributos. O número de colunas por linha é colocado em C pela instrução **ld c,31**.

O apontador HL aumenta, então, em uma unidade, fazendo com que a instrução **ld a,(hl)** coloque no acumulador o conteúdo da segunda posição da tela. Em seguida, o conteúdo do par de registros HL é diminuído em uma unidade, voltando a apontar para a primeira posição da tela. A instrução **ld (hl),a** coloca ali o padrão que antes ocupava a posição seguinte.

O conteúdo do par HL é novamente aumentado, enquanto o conteúdo de C é diminuído. Se o registro C ainda não contém zero — ou seja, se o fim da linha não foi atingido —, a instrução **jr nz,lpj** retorna ao rótulo **lpj** para deslocar o conteúdo da coluna seguinte. Quando o fim da linha for alcançado, a instrução **jr nz,lpj** será ignorada e a instrução **inc hl** aumentará HL em uma unidade.

A instrução **djnz** sempre utiliza o par de registros BC. Como C é igual a zero quando se executa a linha 640 do programa, essa instrução diminui o conteúdo de B em uma unidade e volta ao rótulo **lpi** para deslocar uma nova linha. Depois que a última linha tiver sido deslocada para a direita, B conterá o valor 0, e a instrução **ret** provocará o retorno da sub-rotina.

A ROTINA lg

A primeira instrução dessa sub-rotina é **push bc**, que guarda o contador de colunas na pilha. A instrução **ld bc,15616** coloca em BC o endereço inicial da tabela da ROM em que estão contidos os padrões dos caracteres. O primeiro caractere ali representado é o espaço em branco. Assim, quando a linha 680 chamar a sub-rotina **print**, ela imprimirá um espaço de cor apropriada na tela. A instrução seguinte coloca 32 em DE. A instrução **add hl,de**, por sua vez, soma 32 ao conteúdo do par HL, de modo que ele aponte para o próximo bloco da coluna.

A instrução **pop bc** recupera o contador de colunas e a instrução **djnz** diminui seu valor em uma unidade, retornando ao rótudo **lg** a fim de imprimir mais um espaço em branco. O processo se repete até que o conteúdo de B seja reduzido a zero, indicando que o último bloco foi impresso.

O salto para **lg** não será executado e a instrução **ret** fará com que o processador retorne ao ponto de onde a sub-rotina foi chamada.

T

A rotina Assembly listada a seguir cria e desloca na tela do TRS-Color o cenário de *Avalanche*. Ela difere das demais devido às limitações do uso de cores nesse computador. Como a montanha é coberta de relva e cercada pelo mar azul, não temos alternativa senão pintar o céu de amarelo.

```
10   ORG 19109
20   JSR MODE
30   JSR GCLS
40   LDX #5631
50   LDY #17503
60   LDB #32
70   LOOP PSHS B
80   JSR SCROLL
90   JSR PRINT
100   PULS B
110   DECB
120   BNE LOOP
130   LDY #17604
140   LDX #1569
150   JSR PRSUN
160   RTS
170   MODE EQU 19182
180   GCLS EQU 19161
190   SCROLL EQU 19197
200   PRINT EQU 19218
```

Digite a rotina usando nosso programa Assembler. Grave o programa-fonte e depois monte-o. Grave também a rotina em código, usando o comando **CSAVEM**. A rotina não deve ser executada ainda, pois não funcionará sem os demais programas do artigo.

DESLOCAMENTO DO CENÁRIO

Após termos definido o endereço inicial, precisamos colocar o computador no modo gráfico e selecionar o conjunto de cores que vamos utilizar. Isso é feito pela sub-rotina **MODE**, chamada pelo comando **JSR MODE** na linha 20. A sub-rotina **GCLS** limpa a tela, colorindo-a de amarelo.

A instrução **LDX #5631** coloca em X o endereço do extremo superior esquerdo da tela. Essa área será ocupada pelo último byte do contorno da montanha antes de se iniciar o deslocamento do cenário. A instrução **LDY #17503** coloca em Y o endereço inicial da tabela do perfil da encosta. **LDB #32** coloca em B o número de colunas da tela. A instrução **PSHS B** guarda o contador de colunas na pilha da máquina, liberando o registro B.

A sub-rotina **SCROLL** é chamada para deslocar a coluna uma posição para a direita. Em seguida, a sub-rotina **PRINT** é chamada, imprimindo uma coluna de blocos verdes abaixo do horizonte. A instrução **PULS B**, na linha 100, recupera o contador de colunas da pilha da máquina. O contador é, então, diminuído em uma unidade por **DEC B**. A instrução **BNE LOOP** faz o processador voltar à linha 70 para imprimir uma nova coluna, até que o conteúdo de B acabe se tornando zero.

Quando isso ocorrer, o programa terá desenhado a última coluna, a instrução **BNE LOOP** será ignorada e o programa continuará na linha 140. Ali o registro Y recebe o endereço inicial do padrão do desenho do sol na tabela de blocos gráficos criada no artigo anterior. A linha seguinte coloca em X a posição do sol na tela gráfica. A instrução **JSR PRSUN** chama a sub-rotina que desenha o sol.

Como de costume, **RTS** provoca o retorno da sub-rotina ao interpretador BASIC — ou, quando o jogo estiver completo, ao programa principal.

Várias sub-rotinas chamadas não estão listadas no programa-fonte. Falsas instruções **EQV** informam seus endereços iniciais ao Assembler, para que ele possa calcular os saltos e desvios.

O INEVITÁVEL AMARELO

A rotina **GCLS** limpa a tela, colorindo-a totalmente de amarelo — cor do céu nesta versão do jogo.

```
10   ORG 19161
20   GCLS LDX #1536
30   LDA #85
40   GCLSI STA,X+
50   CMPX #7680
60   BLO GCLSI
70   RTS
```

A rotina começa colocando em X o endereço inicial da tela — 1536. A instrução **LDA #85** coloca no acumulador o código da cor amarela.

A instrução **STA ,X+** coloca o conteúdo do acumulador no endereço apontado por X e aumenta o valor de X em uma unidade. A instrução **CMPX #7680** compara X com 7680, primeiro byte acima do final da tela. A instrução **BLO GOLSI** retorna à linha 40 para colorir a próxima posição de tela, enquanto o final da memória de vídeo não tiver sido alcançado por X — ou seja, enquanto X

for menor que 7680. Quando isso acontece, a instrução **RTS** faz com que o processador volte ao ponto de onde a rotina foi chamada.

MODO GRÁFICO

As quatro sub-rotinas seguintes podem ser montadas juntas, pois ocupam posições consecutivas na memória.

Para modificar o modo gráfico do TRS-Color temos que nos comunicar com dois circuitos integrados especiais que controlam o vídeo. Esses chips são o Gerador de Vídeo ou VDG (*Video Display Generator*) e o Multiplexador Síncrono de Endereços ou SAM (*Synchronous Address Multiplexor*).

Na linha 20, o valor 229 é colocado no acumulador. A seguir, a instrução **STA 65314** coloca o mesmo valor no endereço FF22. Essa posição da memória controla a saída de dados para o VDG e outros periféricos. Cada bit desse byte tem uma função diferente.

Na linha 30, o número 229 — 11100101 — estabelece as linhas de comunicação. O "1" no bit sete determina o modo gráfico; "0" seria o modo de texto. Os bits seis e cinco, valendo "1", definem o modo gráfico P3. O bit três seleciona as cores — aqui, "0" dá verde, vermelho, amarelo e azul.

Os bits dois, um e zero não se referem ao VDG. Eles controlam o tamanho da memória RAM, a produção de sons e a saída para a impressora, respectivamente. Sua conformação usual é 101. Por isso, quando for modificar o conteúdo desse byte, coloque 101 nos três primeiros bits, a menos que haja alguma razão para não fazê-lo.

Quando alteramos as linhas de comunicação com o VDG, devemos fazer também algumas alterações nos bytes que se referem ao SAM. Esse chip tem um registro de dezesseis bits que corresponde às posições de memória que vão de FFCO a FFDF — intervalo que contém 32 bytes. Cada bit do registro existente nessas posições é ativado quando colocamos um valor nos bytes ímpares correspondentes, e apagado quando fazemos o mesmo nos bytes pares. O valor colocado nos bytes e no registro do VDG deve ser o mesmo. Os bits ativados pelas linhas 40 a 60 do programa-fonte informam ao SAM que a memória de vídeo começa no endereço 1536.

```
10  ORG 19182
20  MODE LDA #229
30  STA 65314
40  STA 65475
50  STA 65477
60  STA 65479
70  RTS
```

SCROLL HORIZONTAL

O rótulo **SCROLL** marca o início da rotina que executa um deslocamento horizontal do conteúdo da tela para a direita. Esse processo — chamado geralmente de **SCROLL** horizontal — permite que desenhemos o cenário a partir da extremidade esquerda do vídeo.

```
10  SCROLL PSHS X,Y
20  LDX #1536
30  LDY #1537
40  SCRO LDA ,Y+
50  STA ,X+
60  CMPX #7679
70  BLO SCRO
80  PULS Y,X
90  RTS
```

Essa rotina precisa utilizar os registros X e Y, mas números muito importantes foram guardados ali por outras

rotinas. Assim, a primeira providência do programa, na linha 10, é guardar esses valores na pilha da máquina, com a instrução **PSHS X,Y**.

As linhas 20 e 30 colocam em X e Y os endereços da primeira e da segunda posições da tela, respectivamente. A instrução **LDA ,Y+** coloca em A o conteúdo da posição apontada por Y, e Y aumenta em uma unidade.

A instrução **STA ,X+** coloca o conteúdo de A na posição apontada por X, e X aumenta em uma unidade. Portanto, na primeira passagem do processador, as linhas 40 e 50 colocam o conteúdo da segunda posição da tela dentro da primeira posição, atualizando ainda os valores dos dois apontadores.

A instrução **CMPX #7679** compara o conteúdo de X com o último endereço da memória de vídeo. Enquanto X contiver um valor inferior, a instrução **BLO SCROLL** envia o processador de volta à linha 10 para deslocar mais uma posição de memória para a esquerda.

A rotina não só desloca todo o conteúdo da tela uma posição para a esquerda como também coloca o que havia na última coluna da esquerda para a última coluna da direita, uma linha acima. Isso não tem importância, já que a última coluna da direita será apagada pelos blocos que desenham o cenário.

Quando o conteúdo da última posição da memória de vídeo for deslocado para a esquerda, o conteúdo original dos registros X e Y será recuperado da pilha pelo comando **PULS Y,X**. A instrução **RTS** da linha 90 provocará o retorno da sub-rotina.

O NOVO CENÁRIO

Os blocos gráficos que compõem o cenário preenchem a última coluna da direita, à medida que o conteúdo da tela vai sendo deslocado para a esquerda. A rotina responsável por esse processo é a seguinte:

```
10   PRINT PSHS X
20   LDA ,Y+
30   SUBA #33
40   BNE PRZ
50   PULS X
60   LEAX -256,X
70   PSHS Ⓨ
80   LDY #17536
90   LDB #8
100   PRI LDA ,Y+
110   STA ,X
120   LEAX 32,X
130   DECB
140   BNE PRI
150   PULS Y
160   PRZ CLR ,X
170   LEAX 32,X
180   CMPX #7680
190   BLO PRZ
200   PULS X
```

Desta vez a rotina usará o valor contido no apontador Y. Se estivermos desenhando uma porção inclinada da montanha, poderá ser necessário modificar a altura do horizonte contida no registro X. Mas, por enquanto, a instrução **PSH X** guarda o conteúdo de X na pilha da máquina.

A instrução **LDA ,Y+** coloca em A o conteúdo do endereço indicado por Y e o apontador aumenta em uma unidade. A seguir, a instrução **SUBA #33** subtrai 33 unidades de A.

O número 33 é o código ASCII do ponto de exclamação e, na tabela criada pelo programa do artigo anterior, significa que a encosta é inclinada naquela posição: Se este não for o caso, a subtração não resulta em zero, o que faz a instrução **BNE PRZ** provocar um salto para a linha 160. Se A contiver o valor 33, indicando inclinação da encosta, a instrução **BNE** é ignorada e o programa continua.

A instrução **PULS X** recupera da pilha a altura do horizonte, e a instrução seguinte subtrai 256 unidades de X. O número 256 é igual a 8 × 32 — o apontador X subiu, assim, oito linhas (ou um bloco) na tela. O novo valor de X é recolocado na pilha da máquina para ser usado no desenho da coluna seguinte. O valor do apontador Y — que aponta para a tabela de contorno da montanha — também é guardado na pilha.

A instrução **LDY #17536** coloca em Y o endereço inicial do padrão do bloco de uma porção inclinada da montanha. O bloco é desenhado na posição apontada por X. Neste modo gráfico os pontos são criados em grupos de dois, com base no valor de dois bits. Dois bits podem ter quatro valores diferentes, um para cada cor. A organização da memória de vídeo foi explicada no artigo da página 86.

A instrução **DECB** diminui o conteúdo do contador B em uma unidade e, enquanto esse contador não tiver sido reduzido a zero, a instrução **BNE PRI** fará com que o processador volte à linha 100 para desenhar a próxima linha. Quando B contiver zero, a última linha terá sido desenhada.

Em seguida, a instrução **PULS Y** recupera da pilha o apontador da tabela de contorno da montanha. O programa continua na linha rotulada **PRZ**. Esta é a rotina para a qual o programa saltaria se a montanha fosse plana neste ponto (veja linha 40). Sua função é desenhar uma coluna de blocos verdes, recobrindo parte da montanha.

COMO FUNCIONA

A instrução **CLR ,X** limpa o conteúdo do endereço apontado por X. Limpar — ou seja, tornar igual a zero — o conteúdo de uma posição de memória equivale, no caso, a pintar aquela posição com a cor verde. A instrução **LEAX 32,X** soma 32 ao apontador X, fazendo-o descer uma linha na tela.

A instrução **CMPX #7680** verifica se o apontador ultrapassou o final da memória de vídeo. Enquanto isso não ocorrer, a instrução **BLO PRZ** fará o processador voltar ao rótulo **PRZ** para desenhar mais uma linha de oito pontinhos verdes. Ao se completar a coluna de blocos verdes, a instrução **PULS X** recupera da pilha o valor da altura do horizonte. A instrução **RTS** marca o final da sub-rotina.

O SOL

Os dados necessários ao desenho do sol (padrões e posições na tela) se encontram na tabela criada no artigo anterior. Esses dados são utilizados pela rotina Assembly listada a seguir, que preenche um espaço de 32 por 30 pontos no céu.

```
10   PRSUN LDB #30
20   PRSUNI PSHS B
30   LDB #4
40   PRSUNZ LDA ,Y+
50   STA ,X+
60   DECB
70   BNE PRSUNZ
80   LEAX 28,X
90   PULCS B
100   DECB
110   BNE PRSUNI
120   RTS
```

A instrução **LDB #30** coloca no contador B o número de linhas que serão preenchidas para desenhar o sol. Depois, a instrução **PULS B** guarda o contador na pilha da máquina. A linha seguinte coloca o número 4 no registro B, definindo em quatro bytes — ou 4 × 8 = 32 bits — a largura do desenho. Mesmo que quiséssemos desenhar um sol com 30 × 30 pontos, precisaríamos de um quadriculado de 32 × 30 pontos — usaríamos, no caso, quatro bytes, deixando duas colunas com a cor de fundo.

Como de costume, a instrução **LDA ,Y+** obtém o padrão de uma parte dos desenhos no endereço apontado por Y, colocando-o em A e aumentando em uma unidade o apontador. A instrução **STA ,X+** coloca esse padrão na tela, na posição apontada por X, aumentando também em uma unidade o valor desse apontador. A instrução **DECB** diminui o valor do contador e, enquanto B não for zero, a instrução **BNE PRSUNZ** retorna à linha 40, fazendo com que quatro bytes sejam colocados na tela.

Quando o último byte da linha tiver sido desenhado, a instrução **LEAX 28,X** soma 28 ao conteúdo de X, de modo que ele aponte para o início da próxima linha. O contador de linhas é recuperado da pilha por **PULS B** e diminuído em uma unidade por **DECB**. Enquanto as trinta colunas não tiverem sido desenhadas, a instrução **BNE PRSUNI** faz com que o processador volte à linha 20, repetindo o processo.

Se a última linha foi traçada, B reduz-se a zero, a instrução da linha 110 é ignorada e o programa encontra a instrução **RTS**, que provoca o retorno da sub-rotina.

Quando escrevemos a página de instruções, a tela estava no modo texto de quarenta colunas, totalmente inadequado para o cenário de um jogo de ação. A rotina Assembly a seguir seleciona o modo gráfico de alta resolução.

```
10    org -12144
20    ld hl,62441
30    ld (hl),4
40    inc hl
50    ld (hl),7
60    inc hl
70    ld (hl),7
80    call 114
90    ld a,226
100    ld (62432),a
110    call 105
120    ret
130    end
```

Para selecionar as cores da tela, é preciso modificar o conteúdo de algumas posições da RAM, nas quais o sistema armazena valores usados no controle das diversas operações do micro.

A instrução que se segue à definição do endereço inicial coloca em HL o endereço do byte que determina a cor de frente usada na tela. Depois, a instrução **ld (hl),4** utiliza o endereçamento indireto para colocar ali o código da cor azul-escuro.

O par HL, usado como apontador, tem seu conteúdo aumentado em uma unidade pela instrução **inc hl**, passando a apontar para o próximo endereço, que determina a cor de fundo da tela. Esse byte recebe o valor 7, que é o código da cor ciano. Duas outras instruções colocam o mesmo código na posição seguinte, que determina qual será a cor da moldura da tela.

A instrução **call 114** é responsável pela modificação das cores e do modo de exibição da tela. Ela chama uma sub-rotina do sistema, que coloca o computador no modo gráfico de alta resolução. Essa rotina equivale a um comando **SCREEN 2,0**, ou seja, ela não somente coloca o computador no modo gráfico como também limpa a tela, preenche as tabelas da VRAM com seu conteúdo padrão e, ainda, acerta os conteúdos do VDP (*Video Display Processor*), chip de imagem do MSX.

A rotina chamada na linha 80, contudo, preparou o VDP para exibir sprites pequenos (8 × 8 pontos), quando nós precisamos de sprites grandes (16 × 16 pontos). Para fazer os acertos necessários, teremos que modificar o conteúdo de um dos registros do VDP. Estes podem ser alterados através das posições de memória que vão de 62431 a 62438 — o que corresponde a oito registros de oito bits. As funções de cada registro do VDP serão comentadas em outro artigo. Por hora, interessa-nos apenas o registro 1, que modificaremos através do endereço 62432.

As instruções das linhas 90 e 100 da listagem colocam nesse registro o valor 226. Note que o uso do acumulador é temporário, justificando-se pela inexistência de uma instrução que coloque um número diretamente em uma posição de memória. Não há uma instrução **ld (62432),226**.

Cada bit do registo 1 do VDP tem uma função. O bit 7 indica o tamanho da VRAM utilizada, o bit 6 liga e desliga a tela, o bit 5 seleciona o modo de interrupção, os bits 3 e 4 determinam o tipo de tela (dois bits nos dão quatro opções: telas 0, 1, 2 e 3), o bit 2 não é utilizado, o bit 1 controla o tamanho lógico do sprite (8 × 8 ou 16 × 16 pontos) e o bit 0, finalmente, controla o tamanho físico do sprite (normal ou ampliado). Assim, como 226 é 11110010 em sistema binário, quando colocado no registro 1, ele determina uma VRAM de dezesseis Kbytes, tela ligada, com possibilidade de interrupção, modo gráfico de alta resolução e sprites grandes não ampliados.

A sub-rotina chamada na linha 80 preencheu a tabela de atributos de sprites (veja o artigo da página 808) com nomes compatíveis com sprites pequenos. A instrução **call 105** chama, então, outra sub-rotina da ROM, que preenche a mesma tabela com valores compatíveis com sprites de 16 × 16 pontos. Na realidade, essa sub-rotina preenche a tabela de atributos com seu conteúdo padrão, conforme os valores dos registros do VDP. Depois disso, nossa rotina termina com a instrução **ret**.

TABELA DE PADRÕES

A tela de alta resolução dos microcomputadores da linha MSX é organizada em cinco tabelas: nomes, padrões, cores, atributos de sprites e padrões de sprites. Os efeitos visuais do videogame são criados por intermédio da modifi-

cação dos valores contidos nessas tabelas.

A rotina listada a seguir transfere para a tabela de padrões os blocos gráficos criados pelo programa BASIC do artigo anterior. Como os mesmos blocos são transferidos para a tabela de padrões de sprites, podemos utilizar as figuras que criamos tanto na forma de blocos gráficos como na forma de sprites. Observe que o programa exige que o banco de blocos anteriormente criado esteja na memória.

```
10    org -12121
20    ld de,(62411)
30    ld hl,-15100
40    ld bc,640
50    push bc
60    push hl
70    push de
80    call 92
90    pop de
100    ld hl,2048
110    add hl,de
120    ld d,h
130    ld e,l
140    pop hl
150    pop bc
160    push bc
170    push hl
180    push de
190    call 92
200    pop de
210    ld hl,2048
220    add hl,de
230    ld d,h
240    ld e,l
250    pop hl
260    pop bc
270    push bc
280    push hl
290    call 92
300    ld de,(62415)
310    pop hl
320    pop bc
330    call 92
340    ret
350    end
```

Para transferir os padrões da RAM para a VRAM, utilizamos a sub-rotina da ROM que fica no endereço 92. Usamos a mesma sub-rotina para criar a página inicial e escrever as instruções.

A instrução **ld de,(62411)** coloca em DE o endereço inicial da tabela de padrões. Esse endereço fica armazenado nos bytes 62411 e 62412 e equivale a **BASE(12)**. Note que a instrução transfere dezesseis bits, ou seja, um par de bytes para um par de registros. O registro E recebe o conteúdo de 62411, byte menos significativo. D recebe o byte mais significativo, 62412.

A instrução seguinte coloca em HL o endereço inicial do banco de blocos que criamos na memória. O número de bytes a serem transferidos para a VRAM é colocado em BC por **ld bc,700**.

A sub-rotina que começa no endereço 92 baseia-se no conteúdo dos registros DE, HL e BC para efetuar a transferência. DE deve conter o endereço inicial da porção da VRAM a ser modificada; HL, o endereço inicial da porção da RAM que vai ser transferida; e BC, o número de bytes. Como a sub-rotina altera o conteúdo desses mesmos registros no decorrer de seu funcionamento, eles são guardados na pilha, para eventual uso futuro. As instruções **push bc, push hl** e **push de** tratam disso, antes que **call 92** mande o processador executar a sub-rotina.

A tela de nomes da tela de alta resolução é dividida em três porções, cada uma representando os padrões contidos no terço correspondente da tabela de padrões. Como usaremos toda a tela, precisaremos ter três cópias do banco de blocos na tabela de padrões.

Para fazer a segunda cópia, o endereço inicial da tabela de padrões é recuperado como **pop de**. O par HL recebe, então, o valor 2048, que corresponde à posição inicial do segundo terço da tabela de padrões. Esse valor é somado ao endereço inicial da tabela, para obtermos o endereço inicial do terço médio na VRAM. A instrução que faz esta soma — **add hl,de** — deixa o resultado da operação em HL.

Precisamos do endereço inicial em DE. Como não existem as instruções **add de,hl** e **ld de,hl**, para transferir o resultado para o par HL utilizamos duas instruções: **ld d,h** e **ld e,l**.

O endereço inicial do banco de blocos na RAM é recuperado da pilha com **pop hl** e o número de bytes a serem transferidos para a VRAM, como **pop bc**. Observe que a ordem de recuperação dos pares de registro da pilha é a mesma de sua colocação. Os registros são guardados na pilha novamente, antes que **call 92** volte a transferir o banco de blocos para a VRAM.

Obtém-se a terceira cópia do banco de blocos de maneira muito parecida. O endereço inicial do terço médio é recuperado e somado a 2048, resultando no endereço correspondente ao terço inferior. HL e BC são recuperados da pilha e a rotina da memória ROM é executada novamente. Desta vez, somente BC e HL são guardados na pilha.

Uma quarta cópia do banco de blocos deve ser feita na tabela de padrões de sprites. O endereço inicial da tabela é armazenado nos endereços 62415 e 62416. A instrução **ld de,(62415)** transfere esses dois bytes para DE, como foi explicado anteriormente.

As instruções **pop hl** e **pop bc** recuperam da pilha o endereço inicial e o tamanho do banco. A cópia é feita com **call 92**. A rotina termina com uma instrução **ret**.

A TABELA DE CORES

Se não colocarmos valores adequados na tabela de cores, nenhum bloco gráfico será mostrado no vídeo. Um novo programa BASIC, que coloca os códigos de cores no topo da memória, está listado no final do artigo. Os valores são transferidos para a tabela de cores da tela gráfica por esta rotina:

```
10    org -12066
20    ld de,(62409)
30    ld hl,-16100
40    ld bc,672
50    push bc
60    push hl
70    push de
80    call 92
90    pop de
100    ld hl,2048
110    add hl,de
120    ld d,h
130    ld e,l
140    pop hl
150    pop bc
160    push bc
170    push hl
180    push de
190    call 92
200    pop de
210    ld hl,2048
220    add hl,de
230    ld d,h
240    ld e,l
250    pop hl
260    pop bc
270    call 92
280    ret
290    end
```

Essa rotina é bem parecida com a anterior. Ela começa colocando em DE o endereço inicial da tabela de cores, armazenado nos endereços 62409 e 62410. HL recebe o endereço inicial da lista de códigos de cores que será criada pelo programa BASIC no final do artigo. BC recebe o número de bytes a serem transferidos. Os conteúdos desses três pares de registros são guardados na pilha e a rotina do endereço 92 é chamada novamente.

Utiliza-se o mesmo processo explicado anteriormente para a obtenção de três cópias do banco de cores — cada uma correspondendo a um terço do vídeo. Sprites serão coloridos no momento em que surgirem no vídeo.

SCROLL HORIZONTAL

Para realizar o deslocamento da tela para a direita, usaremos um processo muito semelhante ao que foi descrito no artigo da página 213.

As três rotinas listadas abaixo realizam o processo da translação do conteúdo da tela gráfica:

```
10    org -12022
20    ld hl,(62407)
30    ld de,-6000
40    ld bc,768
50    call 89
60    ret
70    org -12009
80    ld de,-5233
90    ld hl,-5234
100    ld b,24
110    loop push bc
120    ld a,(de)
130    ld bc,31
140    lddr
150    ld (de),a
160    dec hl
170    dec de
180    pop bc
190    djnz loop
200    ret
210    org -11987
220    ld de,(62407)
230    ld hl,-6000
240    ld bc,768
250    call 92
260    ret
270    end
```

A primeira rotina transfere o conteúdo da tabela de nomes da tela de alta resolução para um *buffer* localizado no topo da memória.

O endereço inicial dessa tabela na VRAM fica guardado nos bytes 62407 e 62408 da RAM. A instrução da linha 20 do programa coloca esse valor no par de registros HL. O endereço inicial do buffer é colocado em DE por **ld de,-6000**. O número de bytes a serem transferidos vai para o par de registros BC. A instrução **call 89** chama, então, a sub-rotina da ROM que transfere números da VRAM para a RAM.

A segunda sub-rotina, que começa na linha 70, promove o deslocamento do conteúdo do buffer para a direita.

Os endereços dos dois últimos bytes do buffer são colocados em DE e HL. O número de linhas da tela vai para B. Como BC será utilizado adiante, a instrução **push bc** armazena o contador de linhas na pilha.

O conteúdo da última posição da tela é guardado no acumulador por **ld a,(de)**. O par BC recebe o número de colunas de uma linha — 31. A instrução **lddr** coloca no endereço apontado por DE o conteúdo do endereço apontado por HL e subtrai uma unidade de cada um dos apontadores e do contador BC. O processo é repetido automaticamente até que BC seja zero. A instrução da linha 140 desloca, assim, toda uma linha de blocos para a direita.

O conteúdo da última posição da linha é colocado na primeira posição, agora apontada por DE. As instruções **dec hl** e **dec de** fazem com que esses registros apontem para as duas últimas posições da próxima linha a ser deslocada. No nosso caso, ela fica imediatamente acima da primeira.

O contador de linhas é recuperado da pilha e **djnz** repete o laço **loop** até que todas as 24 linhas de blocos gráficos que compõem o buffer tenham sido deslocadas.

A última rotina, que começa na linha 210, transfere o conteúdo do buffer de volta para a tabela de nomes. Suas instruções são parecidas com as da rotina da linha 10, só que DE recebe o endereço da tabela na VRAM e HL recebe o endereço do buffer. A rotina chamada fica no endereço 92 da ROM.

A ROTINA PRINCIPAL

Para desenhar a montanha que Willie deve escalar, o programa usa a seguinte rotina:

```
10    org 53563
20    ld a,255
30    ld hl,(62407)
40    ld bc,768
50    call 86
60    call -12121
70    call -12066
80    call -12022
90    ld a,8
100    ld (-5230),a
110    ld a,0
120    ld (-5229),a
130    ld hl,-6001
140    ld b,33
150    mp1 push bc
160    push hl
170    call -11987
180    call -12009
190    pop hl
200    ld a,(-5229)
210    cp 1
220    jr nz,tq
230    ld a,(-5230)
240    inc a
250    ld (-5230),a
260    tq ld a,0
270    ld (-5229),a
280    ld a,(hl)
290    dec hl
300    push hl
310    cp 33
320    jr nz,lv
330    ld a,1
340    ld (-5229),a
350    lv ld a,(-5230)
360    ld b,a
370    ld hl,-6000
380    ld a,255
390    call lg
400    ld b,48
410    ld a,(-5229)
420    cp 1
430    jr nz,no
```

```
440    ld b,52
450    no ld a,b
460    ld (hl),a
470    ld a,(-5230)
480    ld b,a
490    ld a,23
500    sub b
510    ld b,a
520    ld a,81
530    ld de,32
540    add hl,de
550    call lg
560    pop hl
570    pop bc
580    djnz mpi
590    ret
600    lg ld (hl),a
610    ld de,32
620    add hl,de
630    djnz lg
640    ret
650    end
```

A tabela de nomes será utilizada para representar blocos gráficos. Estes devem estar devidamente representados nas tabelas de padrões e de cores. Para entender o funcionamento das rotinas gráficas é necessário conhecer a organização da VRAM.

A primeira coisa que a rotina principal faz é preencher a tabela de nomes com o número 255. Isso equivale a encher a tela com o bloco gráfico de código 255, que, no nosso caso, é um bloco ciano. O acumulador recebe o código do bloco, 255. A instrução **ld hl,(62407)** coloca o endereço inicial da tabela de nomes em HL. BC recebe o número de bytes da tabela — 768.

A rotina da memória ROM chamada por **call 86** preenche com o conteúdo de A uma porção da VRAM. O par de registros HL aponta o endereço inicial da porção e o par BC, o número de bytes que sofrerão alterações.

A linha 60 chama a sub-rotina que gera os padrões dos bytes, e a 70, a sub-rotina que cuida da tabela de cores. A sub-rotina do endereço −12022 coloca o conteúdo da tabela de nomes no buffer de deslocamento.

As linhas 90 e 100 colocam no endereço -5230 o número da linha em que colocaremos o horizonte. O endereço -5229 recebe o valor zero. Esse byte servirá de indicador de inclinação da montanha. Se valer zero, ela será plana; se valer um, será inclinada.

O par HL recebe o último valor da tabela de perfil da encosta. Essa tabela será gerada por um programa BASIC listado no final do artigo. O perfil é definido por meio de códigos que indicam se a montanha é plana ou inclinada num determinado local.

Na linha 140, B recebe o número de colunas, que é logo colocado na pilha para liberar o uso desse registro. As linhas 170 e 180 chamam as rotinas que copiam o buffer na tabela de nomes e realizam o deslocamento.

As linhas 200 a 250 verificam se o indicador de inclinação, no endereço -5229, foi modificado anteriormente. Quando isso acontecer, a linha do horizonte deve ser alterada no endereço -5230, o que é feito pelas linhas 230 a 250. A seguir, o indicador recebe novamente o valor zero.

A instrução **ld a,(hl)** coloca em A o valor da tabela de perfil da encosta apontado por HL. Em seguida, esse par de registros é diminuído em uma unidade e guardado na pilha.

O conteúdo de A é comparado a 33 — valor que indica região inclinada. Se A tiver outro valor, a instrução **jr nz,lv** salta para o rótulo **lv**. Caso contrário, as linhas 330 e 340 colocam o número 1 no indicador de inclinação.

A ROTINA lv

As instruções das linhas 350 e 360 colocam em B o número da linha do horizonte. Isso é feito com o auxílio do acumulador, já que não existe uma instrução que coloque o conteúdo de um endereço diretamente em B. HL recebe o endereço inicial do buffer, que corresponde ao topo da primeira coluna da tela. A instrução **ld a,255** coloca o código do bloco ciano no acumulador. Esse bloco será usado para desenhar o céu. A rotina **lg** desenha uma coluna de blocos ciano.

A instrução **ld b,48** coloca em B o código do bloco que representa uma porção plana da montanha. As duas linhas seguintes verificam o conteúdo do indicador de inclinação. Se a encosta for plana nessa coluna, a instrução **jr nz,no** desvia o programa para o rótulo **no**. Se for inclinada, B recebe o código do bloco adequado e a rotina passa ao rótulo **no**.

A ROTINA no

As linhas 450 e 460 colocam o bloco adequado na posição apontada pelo par de registros HL. As duas linhas que se seguem colocam em B o número da linha do horizonte. O registro A recebe o número de linhas da tela e a instrução da linha 500 subtrai o número da linha de horizonte desse valor. Calcula-se, assim, o número de blocos que devem ser colocados abaixo da linha de horizonte nesta coluna.

O resultado da operação fica em A; a instrução **ld b,a** transfere-o para B. A instrução **ld a,81** coloca em A o código do caractere verde, que preenche a encosta. O par de registros DE recebe o número 32 e **add hl,de** soma esse valor ao conteúdo de HL, para que este aponte para a linha seguinte. A rotina **lg** é chamada novamente para desenhar uma coluna de blocos.

Para finalizar, a posição do apontador da tabela de perfil da encosta da montanha é recuperado da pilha por **pop hl**. O contador de colunas também sai da pilha com **pop bc**. A instrução **djnz** repete o processo até que as 32 colunas do desenho tenham sido traçadas e deslocadas na tela.

A ROTINA lg

Essa rotina desenha uma coluna de blocos. O código do bloco deve estar no acumulador; o endereço inicial da coluna no buffer deve estar em HL; o número de linhas, em B.

A linha 600 coloca o código do bloco gráfico no endereço dado por HL. DE recebe o número 32, que é logo somado ao conteúdo de HL. A instrução **djnz** repete o processo de acordo com o conteúdo de B.

AS TABELAS

Este programa cria a tabela de cores na RAM:

```
5 CLEAR 200,-16100
10 FOR I=0 TO 20
20 READ A
30 FOR J=0 TO 31
40 POKE -16100+I*32+J,A
50 NEXT J,I
100 DATA 23,23,23,103,103,247,2
47,23,23,199,199,199,55,55,167,
135,215,151,244,244,51
```

A tabela criada tem 672 bytes. Cada valor na linha **DATA** 100 corresponde a um código de cor, que é repetido 32 vezes, uma para cada linha do bloco gráfico.

O programa a seguir cria a tabela do perfil da encosta:

```
10 FOR I=0 TO 31
20 READ A:POKE -6032+I,A
30 NEXT
100 DATA 33,33,35,35,33,35,35,3
5,33,35,35,33,35,33,35,35,35,35
,33,35,33,35,35,35,35,33,35,35,
33,35,35,35
```

Cada valor da linha **DATA** 100 corresponde a uma coluna do desenho. Um número 33 significa porção inclinada, e um 35, porção plana.

UMA AGENDA ELETRÔNICA (1)

Com nosso programa para calendário e agenda, você poderá planejar seus compromissos diários e manter um registro de datas especiais. Aproveite a oportunidade e organize-se!

Você é daquele tipo de pessoa que sempre se esquece do aniversário da esposa ou só se lembra da hora marcada no dentista quando já é tarde demais? Ou, ainda, que se assusta ao descobrir que uma conta de luz está vencida há mais de um mês?

O programa que apresentamos neste artigo cuidará de todos os seus compromissos. Com ele, você não terá mais desculpas para faltar a encontros ou deixar de pagar suas contas na data certa. Tendo uma impressora, você poderá, inclusive, fazer uma cópia da agenda em papel, para consultá-la sempre que estiver longe do computador.

CALENDÁRIO AUTOMÁTICO

Este programa coloca à sua disposição um calendário ou uma agenda. A primeira opção é a mais simples: mostra um calendário para qualquer mês dos anos de 1753 a 29999. A apresentação do calendário é a usual, com os dias da semana no topo e os dias do mês abaixo. O programa leva em consideração, automaticamente, os anos bissextos, e indica a data do domingo de Páscoa no mês correspondente.

É possível, ainda, imprimir um calendário para o ano todo — alternativa especialmente útil se você pretende colocá-lo em sua escrivaninha ou pendurá-lo na parede.

A AGENDA ELETRÔNICA

Nossa agenda permitirá que você organize melhor seu dia-a-dia, mantendo um registro de todos os seus compromissos. As entradas são feitas sob quatro títulos — Finanças, Encontros, Celebrações e Feriados —, o que torna mais fácil encontrar um determinado tipo de informação.

É muito simples dar entrada à informação e, como vantagem adicional, o programa requer um único registro dos eventos regulares. Assim, aniversários e contas a pagar podem ser anotados uma só vez, no dia correto, em todos os meses ou anos subseqüentes. Quando se digita um dado de Finanças, por exemplo, o programa pergunta se o evento é único (apenas para aquela data específica), mensal, trimestral ou anual. Se você está entrando dados sobre um pagamento mensal, apenas digite **M** para mensal, em seguida o nome **ALUGUEL** e, finalmente, o dia do vencimento. A palavra **ALUGUEL** aparecerá naquele dia em todos os meses seguintes.

A opção Celebrações inclui datas como aniversários de nascimento, casamento, enfim, todo evento que tenha

- CALENDÁRIO MENSAL
- CÁLCULO DE UM EVENTO
- CALENDÁRIO ANUAL
- AGENDA DIÁRIA
- REGISTRO DE EVENTOS
- FINANÇAS, ENCONTROS, CELEBRAÇÕES E FERIADOS
- COMO USAR A AGENDA
- PRIMEIRA PARTE DO PROGRAMA

ocorrência anual. Os Encontros e Feriados são tratados como eventos únicos.

Grave os dados logo depois de entrá-los, usando a opção **GRAVAR**, para evitar que alguma informação se perca. Eles são armazenados em um arquivo à parte, chamado DIÁRIO, e podem ser carregados, corrigidos ou apagados a qualquer momento, o que torna muito fácil a atualização da agenda.

O USO DO PROGRAMA

Depois de digitar alguns dados, você poderá observar o funcionamento da agenda. Se teclar o número do mês e ano, verá todas as entradas para aquele mês, incluindo compromissos financeiros ou celebrações levadas de um mês a outro pelo programa.

As entradas são agrupadas nas diferentes categorias. Antes de exibir determinado grupo na tela, o programa espera que uma tecla seja pressionada. Dessa maneira, nenhuma entrada será retirada sem que você possa vê-la. Se dispuser de uma impressora, copie todas as entradas em papel, sempre que assim julgar necessário.

Se você decidir voltar à opção de calendário e apontar o mesmo mês, poderá destacar as datas correspondentes a compromissos na agenda pressionando, para cada uma das categorias, as teclas $, E, C ou F.

A escolha de uma ou de outra opção do programa depende daquilo que se quer encontrar. Se você pretende examinar a programação para um determinado mês, a escolha mais adequada será a da agenda. Mas você pode precisar exclusivamente de informações sobre os eventos financeiros do ano, para fazer um planejamento. Nesse caso, será melhor selecionar o calendário para um dos meses, destacar os compromissos comerciais teclando $, passar para o mês seguinte e repetir a operação até completar o período desejado.

Como você verá, nosso programa é muito versátil. Você descobrirá várias outras possibilidades tão logo comece a usá-lo.

PRIMEIRA PARTE

Dividimos o programa em três partes. A primeira, que apresentamos a seguir, contém várias rotinas e a tela do menu principal. As outras duas partes, incluindo instruções mais detalhadas de como usar o programa, serão dadas em artigos futuros.

Não se esqueça de que deve gravar esta parte do programa para acrescentá-la às próximas.

```
10 DATA 2,4,1,3,7,31,28,31,30
,31,30,31,31,30,31,30,31
20 DATA "MENS","TRIM","ANUA",
"UNIC"
30 BORDER 0: PAPER 0: INK 7:
CLS
40 CLS
50 CLEAR : LET P=2
60 LET Z$="
                "
70 DIM O$(1,31): DIM Q(4):
DIM L$(4,150,31): DIM T$(4,12
): DIM C(4): DIM Z(5)
80 FOR n=1 TO 5: READ Z(n):
NEXT n
90 DIM D(12): FOR n=1 TO 12:
READ D(n): NEXT n
100 LET M$="Janeiro Fevereiro
Marco    Abril    Maio     Jun
ho    Julho    Agosto   Setemb
ro Outubro  Novembro Dezembro
"
110 LET s$="DomSegTerQuaQuiSex
Sab"
120 DIM P$(4,4): FOR n=1 TO 4:
READ P$(n): NEXT n
130 LET T$(1)="Financas": LET
T$(2)="Apontamentos": LET T$(3
)="Celebracoes": LET T$(4)="Fe
riados"
'140 DEF FN M(A)=((A/K2-INT (A/
K2))*K2)
150 LET SV=0: LET MO=0: LET DA
=0
160 PRINT "HA ALGUMA LISTA DE
DADOS GRAVADA(S/N)  ?": LET K$
="sn"
170 CLS : GOSUB 990: CLS : LET
P=2
180 IF C=1 THEN  GOSUB 760
190 IF C=2 THEN  GOSUB 1760
200 IF C=3 THEN  GOSUB 2240
210 IF C>3 AND C<8 THEN  LET K
B=C-3: GOSUB 1140: LET SV=1
220 IF C=8 THEN  GOSUB 1610:
LET SV=0
230 IF C=9 AND SV=1 THEN
PRINT : PRINT "VOCE NAO GRAVOU
 AS ALTERACOES"'"CONFIRMA A SA
IDA ?": LET K$="sn": GOSUB
1480: IF KB=2 THEN  LET C=0
240 IF C<>9 THEN  GOTO 170
250 CLS : PRINT "ADEUS !"
260 STOP
270 LET MX=0: LET A2=0
280 LET K2=4: IF FN M(YR)=0
THEN  LET A2=1
290 LET K2=100: IF FN M(YR)=0
THEN  LET A2=0
```

```
300 LET K2=400: IF FN M(YR)=0 THEN  LET A2=1
310 IF KB=2 THEN  LET MX=A2+28
320 IF KB=0 THEN  LET KB=1
330 IF KB<>2 THEN  LET MX=D(KB)
340 LET KB=MX: RETURN
350 LET RS=0
360 IF DA=DE AND MO=ME AND KB=5 THEN  LET KB=5: RETURN
370 IF KB=5 THEN  LET KB=7: RETURN
380 LET RS=Z(KB)
390 IF Q(KB)=0 THEN  GOTO 450
400 LET M4=Q(KB)
410 FOR I=1 TO M4
420 LET K$=L$(KB,I): LET K2=3: GOSUB 470: IF VAL K$(2 TO 3)<>DA THEN  LET K2=0
430 IF K2=1 THEN  LET KB=7: RETURN
440 NEXT I
450 LET KB=RS
460 RETURN
470 IF KB<>1 THEN  GOTO 520
480 IF VAL K$(1)=3 THEN  GOTO 540
490 IF VAL K$(1)=4 THEN  GOTO 530
500 IF VAL K$(1)=1 AND VAL K$(2 TO 3)=DA THEN  LET K2=1: RETURN
510 IF VAL K$(1)=2 AND FN M(((YR-VAL K$(6 TO 9))*12)+(12-VAL K$(4 TO 5))+MO))=0 THEN LET K2=1: RETURN
520 IF KB=3 THEN  GOTO 540
530 IF VAL K$(2 TO 3)=DA AND VAL K$(4 TO 5)=MO AND VAL K$(6 TO 9)=YR THEN  LET K2=1: RETURN
540 IF VAL K$(4 TO 5)=MO THEN LET K2=1: RETURN
550 LET K2=0: RETURN
560 LET Y2=0: LET D2=0: LET M2=0
570 LET Y2=YR-1
580 LET D2=Y2*365+INT (Y2/4)-INT (Y2/100)+INT (Y2/400)
590 IF MO=1 THEN  GOTO 630
600 FOR m=1 TO MO-1
610 LET KB=m: GOSUB 270: LET D2=D2+KB
620 NEXT m
630 LET KB=D2+DA: RETURN
640 LET MS=MO: LET DS=DA
650 LET DA=1: LET MO=3: GOSUB 560: LET K2=7: LET DE=FN M(KB)
660 LET N2=(INT (YR/100))-16: LET C2=3+N2-INT ((N2+1)/3)-INT (N2/4)
670 LET K2=19: LET N2=FN M(YR+1): LET K2=30: LET D2=FN M(C2+(N2*19))
680 IF N2>11 AND D2<27 THEN LET D2=D2-1: GOTO 700
690 IF N2<=11 AND D2=29 THEN LET D2=28
700 LET D2=D2+21
710 LET D2=D2+1: LET K2=7: IF INT (FN M(D2+DE)+0.1)<>1 THEN GOTO 710
720 IF D2<32 THEN  LET ME=3
730 IF D2>=32 THEN  LET D2=D2-31: LET ME=4
740 LET DE=INT (D2+0.1): LET MO=MS: LET DA=DS
750 RETURN
760 GOSUB 2510: GOSUB 2480
770 CLS
780 LET MK=5
790 CLS
800 PRINT AT 17,0;"<BREAK> retorna ao menu"
810 PRINT "Teclas z,x alteram Mes"
820 PRINT  INK 7;T$(1); INK 2;" F  "; INK 7;T$(2); INK 4;" A "
830 PRINT  INK 7;T$(3); INK 1;" C  "; INK 7;T$(4); INK 3;" F "
840 PRINT AT 0,0;
850 GOSUB 2570: IF MK<5 THEN PRINT T$(MK)
860 PRINT #P: LET KB=1: GOSUB 1920
870 IF P=3 THEN  PRINT #P
880 PRINT #P: LET T2=MK: LET S2=1: GOSUB 2020
890 LET P=2
900 LET K$="zxfacf ": GOSUB 1480: LET A=KB
 910 IF A=1 THEN  LET MO=MO-1
 920 IF A=2 THEN  LET MO=MO+1
 930 IF MO=13 THEN  LET MO=1: LET YR=YR+1: GOSUB 640
 940 IF MO=0 THEN  LET MO=12: LET YR=YR-1: GOSUB 640
 950 IF A>2 AND A<7 THEN  LET MK=A-2
 960 IF A<3 THEN  LET MK=5
 970 IF A<>7 THEN  GOTO 790
 980 RETURN
 990 CLS : PRINT  PAPER 5; INK 1;AT 0,5;" CALENDARIO & DIARIO "; PAPER 6; INK 0;AT 2,7;" MENU PRINCIPAL "
1000 FOR Z=1 TO 19: PRINT  PAPER 1;"
      ": NEXT Z: PRINT AT 3,0
1010 PAPER 1: INK 7
1020 PRINT AT 4,1;"1- Consultar calendario mensal"
1030 PRINT AT 6,1;"2- Consultar calendario anual"
1040 PRINT AT 8,1;"3- Consultar diario"
```

```
1050 PRINT AT 10,1;"4- Rever/Ed
itar Financas"
1060 PRINT AT 12,1;"5- Rever/Ed
itar Apontamentos"
1070 PRINT AT 14,1;"6- Rever/Ed
itar Celebracoes"
1080 PRINT AT 16,1;"7- Rever/Ed
itar Feriados"
1090 PRINT AT 18,1;"8- Gravar a
s listas"
1100 PRINT AT 20,1;"9- Sair do
programa"
1110 PRINT TAB 9;"Faca a opcao"
```

```
10 CLS
20 CLEAR 5000
30 DIM LI$(3,150),TY$(3),CO(4)
40 DM$="312831303130313130313031
1"
50 MN$="JANEIRO  FEVEREIROMARCO
    ABRIL    MAIO     JUNHO
JULHO    AGOSTO   SETEMBRO OUTU
BRO  NOVEMBRO DEZEMBRO "
60 DN$="DOMSEGTERQUAQUISEXSAB"
70 PA$="MENSTRIMANUAUNIC"
80 TY$(0)="FINANCAS":TY$(1)="EN
CONTROS":TY$(2)="CELEBRACOES":T
Y$(3)="FERIADOS"
90 DEF FNM(A)=INT((A/K2-INT(A/K
2))*K2+0.5)*SGN(A/K2)
100 SV=0:P=0:MO=0:DA=0
110 PRINT @256,"HA ALGUMA LISTA
 DE DAOS GRAVADA? (S/N)"
120 REM
130 CLS:GOSUB 1030:CLS:P=0
140 IF C=1 GOSUB 770
150 IF C=2 GOSUB 2010
160 IF C=3 GOSUB 2460
170 IF C>3 AND C<8 THEN KB=C-4:
GOSUB 1180:SV=1
180 IF C=8 GOSUB 1730:SV=0
190 IF C=9 AND SV=1 THEN PRINT:
PRINT"VOCE NAO GRAVOU AS ALTERA
COES":PRINT"CONFIRMA A SAIDA?":
KB$="SN":GOSUB 1590:IF KB=2 THE
N C=0
200 IF C<>9 THEN 120
210 CLS:PRINT" ADEUS !"
220 END
230 'COMPRIMENTO DO MES
240 MX=0:A2=0
250 K2=4:IF FNM(YR)=0 THEN A2=1
260 K2=100:IF FNM(YR)=0 THEN A2
=0
270 K2=400:IF FNM(YR)=0 THEN A2
=1
280 IF KB=2 THEN MX=A2+28
290 IF KB<>2 THEN MX=VAL(MID$(D
M$,KB*2-1,2))
300 KB=MX:RETURN
310 'CARACTER DECISORIO
320 A3$="":N3=0:F3=0:M3=0
330 IF P=2 THEN RS=32:GOTO 460
340 IF KB=5 AND MO=ME AND DA=DE
 THEN RS=191:GOTO 460
350 IF KB=5 THEN RS=143:GOTO 46
0
360 M3=VAL(LI$(KB,0))
370 REM
380 N3=N3+1
390 A3$=LI$(KB,N3)
400 IF MID$(A3$,2,1)="" THEN D=
0 ELSE D=ASC(MID$(A3$,2,1))
410 IF D<>DA THEN 430
420 KB$=A3$:GOSUB 470:F3=K2
430 IF NOT(F3=1 OR N3>M3)THEN 3
70
440 IF F3=0 THEN RS=32:GOTO 460
450 N3=159+16*KB:RS=N3
460 KB=RS:RETURN
470 'CONFERIR ITEM NO MES
480 T4=0:Y4=0:F4=0
490 T4=ASC(MID$(KB$,1,1))
500 Y4=ASC(MID$(KB$,3,1))
510 M4=(Y4 AND 15)
520 Y4=(FIX(Y4/16)+17)*100+ASC(
MID$(KB$,4,1))
530 IF Y4>YR THEN K2=0:RETURN
540 K2=3:IF(T4=1 AND MO>=M4)OR(
T4=2 AND FNM(M4-MO)=0)OR(T4=3 A
ND M4=MO)OR(T4=4 AND M4=MO AND
Y4=YR) THEN F4=1
550 K2=F4:RETURN
560 'NUMERO DO DIA
570 YX=0:D2=0:M2=0
580 Y2=YR-1
590 D2=Y2*365+FIX(Y2/4)-FIX(Y2/
100)+FIX(Y2/400)
600 IF MO=1 THEN 640
610 FOR M2=1 TO MO-1
620 KB=M2:GOSUB 230:D2=D2+KB
630 NEXT
640 KB=D2+DA:RETURN
650 REM
660 N2=0:C2=0:D2=0
670 MS=MO:DS=DA
680 DA=1:MO=3:GOSUB 560:K2=7:DE
=FNM(KB)
690 N2=FIX(YR/100)-16:C2=3+N2-F
IX((N2+1)/3)-FIX(N2/4)
700 K2=19:N2=FNM(YR+1):K2=30:D2
=FNM(C2+(N2*19))
710 IF N2>11 AND D2<27 THEN D2=
D2-1 ELSE IF N2<=11 AND D2=29 T
HEN D2=28
720 D2=D2+21
730 D2=D2+1:K2=7:IF FNM(D2+DE)<
>1 THEN 730
740 IF D2<32 THEN ME=3 ELSE D2=
D2-31:ME=4
750 DE=D2:MO=MS:DA=DS
760 RETURN
770 'VER CAL.MES
780 REM
790 GOSUB 2750:GOSUB 2720
800 CLS
810 PRINT"SETAS UP/DOWN ALTERAM
 MES"
820 PRINT"USE clear PARA VOLTAR
 AO MENU"
830 PRINT:PRINT CHR$(159);TY$(0
);"($)",CHR$(175);TY$(1)
840 PRINT:PRINT CHR$(191);TY$(2
),CHR$(207);TY$(3)
850 PRINT:PRINT"QUALQUER TECLA
PARA CONTINUAR"
855 IF INKEY$="" THEN 855
```

```
860 MK=5
870 REM
880 CLS
890 GOSUB 2820:IF MK<4 THEN PRI
NT @19,TY$(MK)
900 PRINT #-P:KB=1:GOSUB 2150
910 IF P=2 THEN PRINT #-P
920 PRINT#-P:T2=MK:S2=1:GOSUB 2
240
930 P=0
940 KB$="^"+CHR$(10)+"$ECF"+CHR
$(12):GOSUB 1590:A=KB
950 IF A=1 THEN MO=MO-1
960 IF A=2 THEN MO=MO+1
970 IF MO=13 THEN MO=1:YR=YR+1:
GOSUB 650
980 IF MO=0 THEN MO=12:YR=YR-1:
GOSUB 650
990 IF A>2 AND A<7 THEN MK=A-3
1000 IF A<3 THEN MK=5
1010 IF A<>7 THEN 870
1020 RETURN
1030 'RETORNO AO MENU
1040 PRINT " PROGRAMA  CALENDAR
IO & DIARIO  ";STRING$(32,131)
1050 PRINT TAB(13);"menu"
1070 PRINT"1- CONSULTAR CALENDA
RIO MENSAL"
1080 PRINT"2- CONSULTAR CALENDA
RIO ANUAL"
1090 PRINT"3- CONSULTAR DIARIO"
1100 PRINT"4- REVER/EDITAR FINA
NCAS"
1110 PRINT"5- REVER/EDITAR ENCO
NTROS"
1120 PRINT"6- REVER/EDITAR CELE
BRACOES"
1130 PRINT"7- REVER/EDITAR FERI
ADOS"
1140 PRINT"8- GRAVAR AS LISTAS"
1150 PRINT"9- SAIR DO PROGRAMA"
1160 PRINT:PRINT TAB(9);"FACA A
 OPCAO"
1170 KB$="123456789":GOSUB 1590
:C=KB:RETURN
```

```
10 CLS:COLOR 15,4,4:KEYOFF
12 CLEAR 7000:MAXFILES=3
15 OPEN "CRT:" FOR OUTPUT AS #3
20 OPEN "LPT:" FOR OUTPUT AS #2
30 DIM LI$(3,150),TY$(3),CO(4)
40 DM$="312831303130313130313031
1"
50 MN$="JANEIRO  FEVEREIROMARÇO
   ABRIL    MAIO     JUNHO
JULHO    AGOSTO   SETEMBRO OUTU
BRO  NOVEMBRO DEZEMBRO ":' O nú
mero de espaços deve completar
9 caracteres para cada mês
60 DN$="DOMSEGTERQUAQUISEXSAB"
70 PA$="menstrimanuaunic"
80 TY$(0)="Finanças":TY$(1)="En
contros":TY$(2)="Celebrações":T
Y$(3)="Feriados"
90 DEF FNM(A)=INT((A/K2-INT(A/K
2))*K2+.5)*SGN(A/K2)
100 SV=0:P=0:MO=0:DA=0
110 LOCATE 1,5:PRINT"Você tem l
istas de dados? (S/N)";
120 REM
130 CLS:GOSUB 1030:CLS:P=3
140 IF C=1 THEN GOSUB 770
150 IF C=2 THEN GOSUB 2010
160 IF C=3 THEN GOSUB 2460
170 IF C>3 AND C<8 THEN KB=C-4:
GOSUB 1180:SV=1
180 IF C=8 THEN GOSUB 1730:SV=0
190 IF C=9 AND SV=1 THEN PRINT:
PRINT"Você não gravou as ultima
s alterações!":PRINT"Confirma a
 saída? (S/N)";:KB$="SN":GOSUB
1590:IF KB=2 THEN C=0
200 IF C<>9 THEN 120
210 CLS:CLOSE:PRINT"Até logo...
"
220 END
230 ' Tamanho do mês
240 MX=0:A2=0
250 K2=4:IF FNM(YR)=0 THEN A2=1
260 K2=100:IF FNM(YR)=0 THEN A2
=0
270 K2=400:IF FNM(YR)=0 THEN A2
=1
280 IF KB=2 THEN MX=A2+28
290 IF KB<>2 THEN MX=VAL(MID$(D
M$,KB*2-1,2))
300 KB=MX:RETURN
310 ' Marcar dia de compromisso
320 A3$="":N3=0:F3=0:M3=0
330 IF P=2 THEN RS=32:GOTO 460
340 IF KB=5 AND MO=ME AND DA=DE
 THEN RS=42:GOTO 460
350 IF KB=5 THEN RS=32:GOTO 460
360 M3=VAL(LI$(KB,0))
370 REM
380 N3=N3+1
390 A3$=LI$(KB,N3)
400 IF MID$(A3$,2,1)="" THEN D=
0 ELSE D=ASC(MID$(A3$,2,1))
410 IF D<>DA THEN 430
420 KB$=A3$:GOSUB 470:F3=K2
430 IF NOT (F3=1 OR N3>M3) THEN
 370
440 IF F3=0 THEN RS=32:GOTO 460
450 RS=62
460 KB=RS:RETURN
470 ' Procura por ítem em mês
480 T4=0:Y4=0:F4=0
490 T4=ASC(MID$(KB$,1,1))
500 Y4=ASC(MID$(KB$,3,1))
510 M4=(Y4 AND 15)
520 Y4=(FIX(Y4/16)+17)*100+ASC(
MID$(KB$,4,1))
530 IF Y4>YR THEN K2=0:RETURN
540 K2=3:IF (T4=1 AND MO>=M4) O
R (T4=2 AND FNM(M4-MO)=0) OR (T
4=3 AND M4=MO) OR (T4=4 AND M4=
MO AND Y4=YR) THEN F4=1
550 K2=F4:RETURN
560 ' Det número do dia
570 YX=0:D2=0:M2=0
580 Y2=YR-1
590 D2=Y2*365+FIX(Y2/4)-FIX(Y2/
100)+FIX(Y2/400)
600 IF MO=1 THEN 640
```

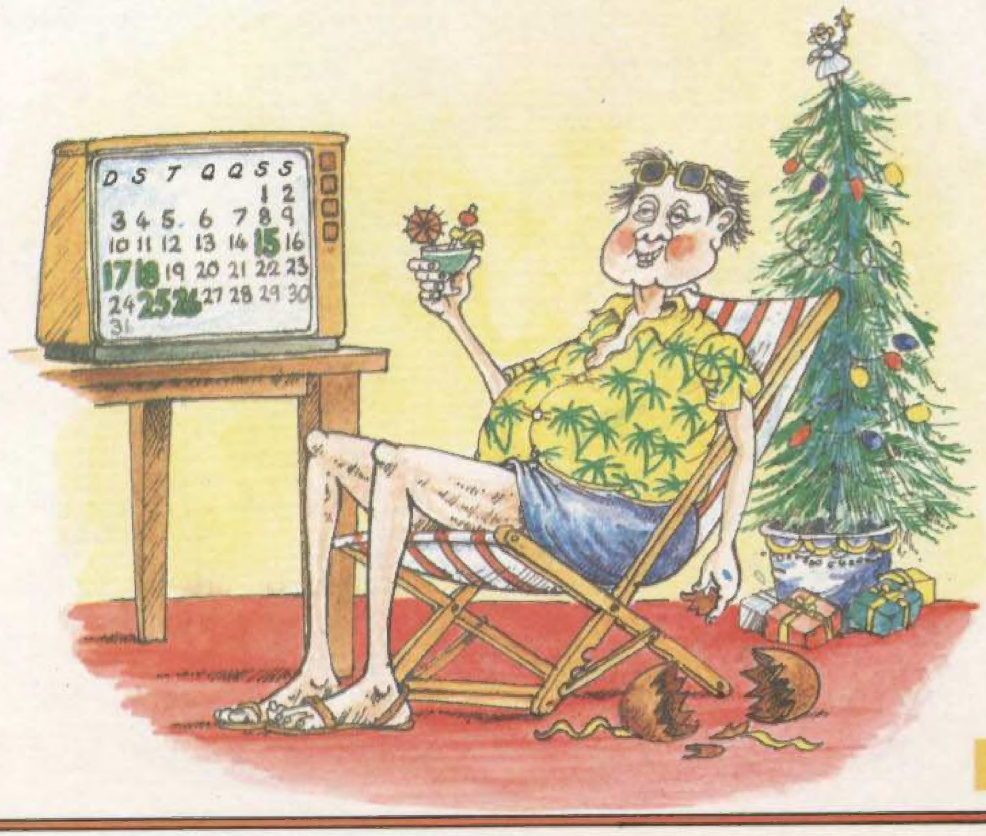
```
610 FOR M2=1 TO MO-1
620 KB=M2:GOSUB 230:D2=D2+KB
630 NEXT
640 KB=D2+DA:RETURN
650 ' Encontrar o dia de Páscoa
660 N2=0:C2=0:D2=0
670 MS=MO:DS=DA
680 DA=1:MO=3:GOSUB 560:K2=7:DE
=FNM(KB)
690 N2=FIX(YR/100)-16:C2=3+N2-F
IX((N2+1)/3)-FIX(N2/4)
700 K2=19:N2=FNM(YR+1):K2=30:D2
=FNM(C2+(N2*19))
710 IF N2>11 AND D2<27 THEN D2=
D2-1 ELSE IF N2<=11 AND D2=29 T
HEN D2=28
720 D2=D2+21
730 D2=D2+1:K2=7:IF FNM(D2+DE)<
>1 THEN 730
740 IF D2<32 THEN ME=3 ELSE D2=
D2-31:ME=4
750 DE=D2:MO=MS:DA=DS
760 RETURN
770 ' Consulta calend mensal
780 REM
790 GOSUB 2750:GOSUB 2720:MK=5
800 CLS:LOCATE 0,20
810 PRINT"<- e -> mudam o mês "
;
820 PRINT"<esc> volta ao menu";
830 PRINT"$:";TY$(0),"E:";TY$(1
)
840 PRINT"C:";TY$(2),"F:";TY$(3
)
850 LOCATE 0,0
860 REM
870 REM
880 REM
890 GOSUB 2820:IF MK<4 THEN LOC
ATE 3,2:PRINTTY$(MK)
895 REMIF P=2 THEN LPRINT
900 PRINT#P,:KB=1:GOSUB 2150
910 IF P=2 THEN LPRINT
920 PRINT#P,:T2=MK:S2=1:GOSUB 2
240
930 P=3
940 KB$=CHR$(29)+CHR$(28)+"$ECF
"+CHR$(27):GOSUB 1590:A=KB
950 IF A=1 THEN MO=MO-1
960 IF A=2 THEN MO=MO+1
970 IF MO=13 THEN MO=1:YR=YR+1:
GOSUB 650
980 IF MO=0 THEN MO=12:YR=YR-1:
GOSUB 650
990 IF A>2 AND A<7 THEN MK=A-3
1000 IF A<3 THEN MK=5
1010 IF A<>7 THEN 800
1020 RETURN
1030 ' Menu devolve escolha em
KB
1040 LOCATE10,0:PRINT"CALENDARI
O E DIARIO"
1050 PRINT:PRINTTAB(16);"Menu"
1060 PRINT
1070 PRINT"1- Consultar calendá
rio mensal"
1080 PRINT:PRINT"2- Consultar c
alendário anual"
1090 PRINT:PRINT"3- Consultar d
iário"
1100 PRINT:PRINT"4- Rever/edita
r finanças"
1110 PRINT:PRINT"5- Rever/edita
r encontros"
1120 PRINT:PRINT"6- Rever/edita
r celebrações"
1130 PRINT:PRINT"7- Rever/edita
r feriados"
1140 PRINT:PRINT"8- Gravar as l
istas"
1150 PRINT:PRINT"9- Fim de prog
rama"
1160 PRINT:PRINTTAB(9);"Escolha
:";
1170 KB$="123456789":GOSUB 1590
:C=KB:RETURN
```

```
10  HOME
30  DIM LI$(3,150),TY$(3),CO(4)

40 DM$ = "312831303130313130313
031"
50 MN$ = "JANEIRO  FEVEREIROMAR
CO    ABRIL    MAIO     JUNHO
  JULHO    AGOSTO   SETEMBRO OU
TUBRO  NOVEMBRO DEZEMBRO "
60 DN$ = "DOMSEGTERQUAQUISEXSAB
"
70 PA$ = "MENSALTRIMESANUAL UNI
CO ":D$ =  CHR$ (13) +  CHR$ (4
)
80 TY$(0) = "FINANCAS":TY$(1) =
 "ENCONTROS":TY$(2) = "CELEBRAC
OES":TY$(3) = "FERIADOS"
90  DEF  FN M(A) =  INT ((A / K
2 -  INT (A / K2)) * K2 + .5) *
  SGN (A / K2)
100 SV = 0:P = 0:MO = 0:DA = 0
110  VTAB 8: PRINT "VOCE TEM LI
STAS DE DADOS? (S/N) ";
120  REM
130  HOME : GOSUB 1030: HOME :P
 = 0
140  IF C = 1 THEN  GOSUB 770
150  IF C = 2 THEN  GOSUB 2010
160  IF C = 3 THEN  GOSUB 2460
170  IF C > 3 AND C < 8 THEN KB
 = C - 4: GOSUB 1180:SV = 1
180  IF C = 8 THEN  GOSUB 1730:
SV = 0
190  IF C = 9 AND SV = 1 THEN
PRINT : PRINT "VOCE NAO GRAVOU
AS ALTERACOES!": PRINT "CONFIRM
A A SAIDA? (S/N)":KB$ = "SN": G
OSUB 1590: IF KB = 2 THEN C = 0

200  IF C <  > 9 THEN 120
210  HOME : PRINT "ATE LOGO..."

220  END
230  REM  DURACAO DO MES
240 MX = 0:A2 = 0
250 K2 = 4: IF  FN M(YR) = 0 TH
EN A2 = 1
260 K2 = 100: IF  FN M(YR) = 0
THEN A2 = 0
270 K2 = 400: IF  FN M(YR) = 0
THEN A2 = 1
280  IF KB = 2 THEN MX = A2 + 2
8
290  IF KB <  > 2 THEN MX =  VA
L ( MID$ (DM$,KB * 2 - 1,2))
300 KB = MX: RETURN
310  REM  CARACTER MARCADOR
320 A3$ = "":N3 = 0:F3 = 0:M3 =
 0
330  IF P = 2 THEN RS = 32: GOT
O 460
340  IF KB = 5 AND MO = ME AND
DA = DE THEN RS = 42: GOTO 460
350  IF KB = 5 THEN RS = 32: GO
TO 460
```



```
360 M3 =  VAL (LI$(KB,0))
370  REM
380 N3 = N3 + 1
390 A3$ = LI$(KB,N3)
400  IF  MID$ (A3$,2,1) = "" TH
EN D = 0
405  IF  MID$ (A3$,2,1) <  > ""
 THEN D =  ASC ( MID$ (A3$,2,1)
)
410  IF (D <  > DA) THEN 430
420 KB$ = A3$: GOSUB 470:F3 = K
2
430  IF  NOT (F3 = 1 OR N3 > M3
) THEN 370
440  IF F3 = 0 THEN RS = 32: GO
TO 460
450 RS = 62
460 KB = RS: RETURN
470  REM  VERIFICA ITEM EM MES
480 T4 = 0:Y4 = 0:F4 = 0
490 T4 =  ASC ( MID$ (KB$,1,1))

500 Y4 =  ASC ( MID$ (KB$,3,1))
:T = Y4
510 Y4 = ( INT ( ABS (Y4 / 16))
 + 17) * 100 +  ASC ( MID$ (KB$
,4,1))
520 M4 = T - (( INT (Y4 / 100)
- 17) * 16)
530  IF Y4 > YR THEN K2 = 0: RE
TURN
540 K2 = 3: IF (T4 = 1 AND ((Y4
 < YR) OR (MO >  = M4 AND Y4 =
YR))) OR (T4 = 2 AND  FN M(M4 -
 MO) = 0) OR (T4 = 3 AND M4 = M
O) OR (T4 = 4 AND M4 = MO AND Y
4 = YR) THEN F4 = 1
550 K2 = F4: RETURN
560  REM  NUMERO DO DIA
570 YX = 0:D2 = 0:M2 = 0
580 Y2 = YR - 1
590 D2 = Y2 * 365 +  INT ( ABS
(Y2 / 4)) -  INT ( ABS (Y2 / 10
0)) +  INT ( ABS (Y2 / 400))
600  IF MO = 1 THEN 640
610  FOR M2 = 1 TO MO - 1
620 KB = M2: GOSUB 230:D2 = D2
+ KB
630  NEXT
640 KB = D2 + DA: RETURN
650  REM  DATA DA PASCOA
660 N2 = 0:C2 = 0:D2 = 0
670 MS = MO:DS = DA
680 DA = 1:MO = 3: GOSUB 560:K2
 = 7:DE =  FN M(KB)
690 N2 =  INT ( ABS (YR / 100))
 - 16:C2 = 3 + N2 -  INT ( ABS
((N2 + 1) / 3)) -  INT ( ABS (N
2 / 4))
700 K2 = 19:N2 =  FN M(YR + 1):
K2 = 30:D2 =  FN M(C2 + (N2 * 1
9))
710  IF N2 > 11 AND D2 < 27 THE
N D2 = D2 - 1: GOTO 720
715  IF N2 <  = 11 AND D2 = 29
THEN D2 = 28
720 D2 = D2 + 21
730 D2 = D2 + 1:K2 = 7: IF  FN
M(D2 + DE) <  > 1 THEN 730
740  IF D2 < 32 THEN ME = 3: GO
TO 750
745 D2 = D2 - 31:ME = 4
750 DE = D2:MO = MS:DA = DS
760  RETURN
770  REM  CALENDARIO MENSAL
780  REM
790  GOSUB 2750: GOSUB 2720:MK
= 5
800  HOME
810  VTAB 21: PRINT "<- ->: MUD
A O MES    ";
820  PRINT "<ESC> = MENU"
830  PRINT "$:"TY$(0),"E:"TY$(1
)
840  PRINT "C:"TY$(2),"F:"TY$(3
);
850  REM
855  REM
870  REM
880  PRINT D$;"PR#";P: VTAB 1:
HTAB 1
890  GOSUB 2820: IF MK < 4 THEN
  HTAB 20: VTAB 1: PRINT TY$(MK
)
900  PRINT : PRINT :KB = 1: GOS
UB 2150
910  IF P = 1 THEN  PRINT
920  PRINT :T2 = MK:S2 = 1: GOS
UB 2240
930 P = 0: PRINT D$;"PR#";P
940 KB$ =  CHR$ (8) +  CHR$ (21
) + "$ECF" +  CHR$ (27): GOSUB
1590:A = KB
950  IF A = 1 THEN MO = MO - 1
960  IF A = 2 THEN MO = MO + 1
970  IF MO = 13 THEN MO = 1:YR
= YR + 1: GOSUB 650
980  IF MO = 0 THEN MO = 12:YR
= YR - 1: GOSUB 650
990  IF A > 2 AND A < 7 THEN MK
 = A - 3
1000  IF A < 3 THEN MK = 5
1010  IF A <  > 7 THEN 800
1020  RETURN
1030  REM  MENU DEVOLVE ESCOLHA
 EM KB
1040  INVERSE : PRINT  TAB( 4)"
PROGRAMA DE CALENDARIO E DIARIO
   ": NORMAL
1050  PRINT : PRINT  TAB( 18)"M
ENU"
1060  PRINT
1070  PRINT  TAB( 8);"1:-VER CA
LENDARIO MENSAL"
1080  PRINT : PRINT  TAB( 8);"2
:-VER CALENDARIO ANUAL"
1090  PRINT : PRINT  TAB( 8);"3
:-VER DIARIO"
1100  PRINT : PRINT  TAB( 8);"4
:-REVER/EDITAR FINANCAS"
1110  PRINT : PRINT  TAB( 8);"5
:-REVER/EDITAR ENCONTROS"
1120  PRINT : PRINT  TAB( 8);"6
:-REVER/EDITAR CELEBRACOES"
1130  PRINT : PRINT  TAB( 8);"7
:-REVER/EDITAR FERIADOS"
1140  PRINT : PRINT  TAB( 8);"8
:-GRAVAR AS LISTAS"
1150  PRINT : PRINT  TAB( 8);"9
:-SAIR DO PROGRAMA"
1160  PRINT : PRINT : PRINT  TA
B( 15)"ESCOLHA ";
1170 KB$ = "123456789": GOSUB 1
590:C = KB: RETURN
```


MICRO DICAS

COMO CALCULAR DATAS EM UM PROGRAMA DE CALENDÁRIO

Um problema sério de programação, para quem deseja desenvolver aplicações que envolvem cálculos de datas ou a determinação das funções de um calendário universal, é a realização desses cálculos em BASIC ou outra linguagem de alto nível.

Há várias soluções para o problema — umas mais fáceis e intuitivas e outras mais difíceis, mas que utilizam algoritmos ("truques" matemáticos de cálculo) bem compactos. Desenvolvido o procedimento para todos os tipos de cálculo, organize-os na forma de sub-rotinas, que poderão ser usadas em diversos programas.

Esse assunto será tratado em um artigo especial, mas adiantamos aqui algumas considerações importantes:

• Teste de datas

Tenha sempre o cuidado de testar a data fornecida pelo usuário, que deverá ser entrada no formato numérico (no Brasil, DD/MM/AAAA).

Verifique se o número do mês cai entre 01 e 12 e se o dia do mês cai entre 1 e o número de dias no mês (28, 29, 30 ou 31). Pode-se armazenar o número de dias para cada mês em um conjunto de doze elementos.

Teste, também, se o ano é bissexto. Para isso, basta verificar se ele é divisível exatamente por 4.

A maneira de testar o ano varia com a aplicação. Se se pede ao usuário a data atual, por exemplo, pode-se incluir na rotina um teste que indique se ela é ou não anterior à data de criação do programa!

• Tipos de data

O tipo de data que utilizamos (DD/MM/AA), chamado de data gregoriana, é muito inconveniente para entrada em computador e para cálculos de diferenças de datas. Para colocá-las em ordem crescente é preciso alterar a ordem dos seus três elementos: o algoritmo de ordenação deve testá-las como AAMMDD.

Existem outros tipos de data, mais racionais. A data ordinal, por exemplo, torna bem mais fácil calcular diferenças entre datas. É expressa numa forma como esta: 123/1986 — ou seja, dia n.º 123 (desde 1.º de janeiro) do ano de 1986. A data fiscal, por sua vez, leva em consideração o número do dia da semana e o número da semana (3/25/1986 é a terça-feira da 25.ª semana de 1986).

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

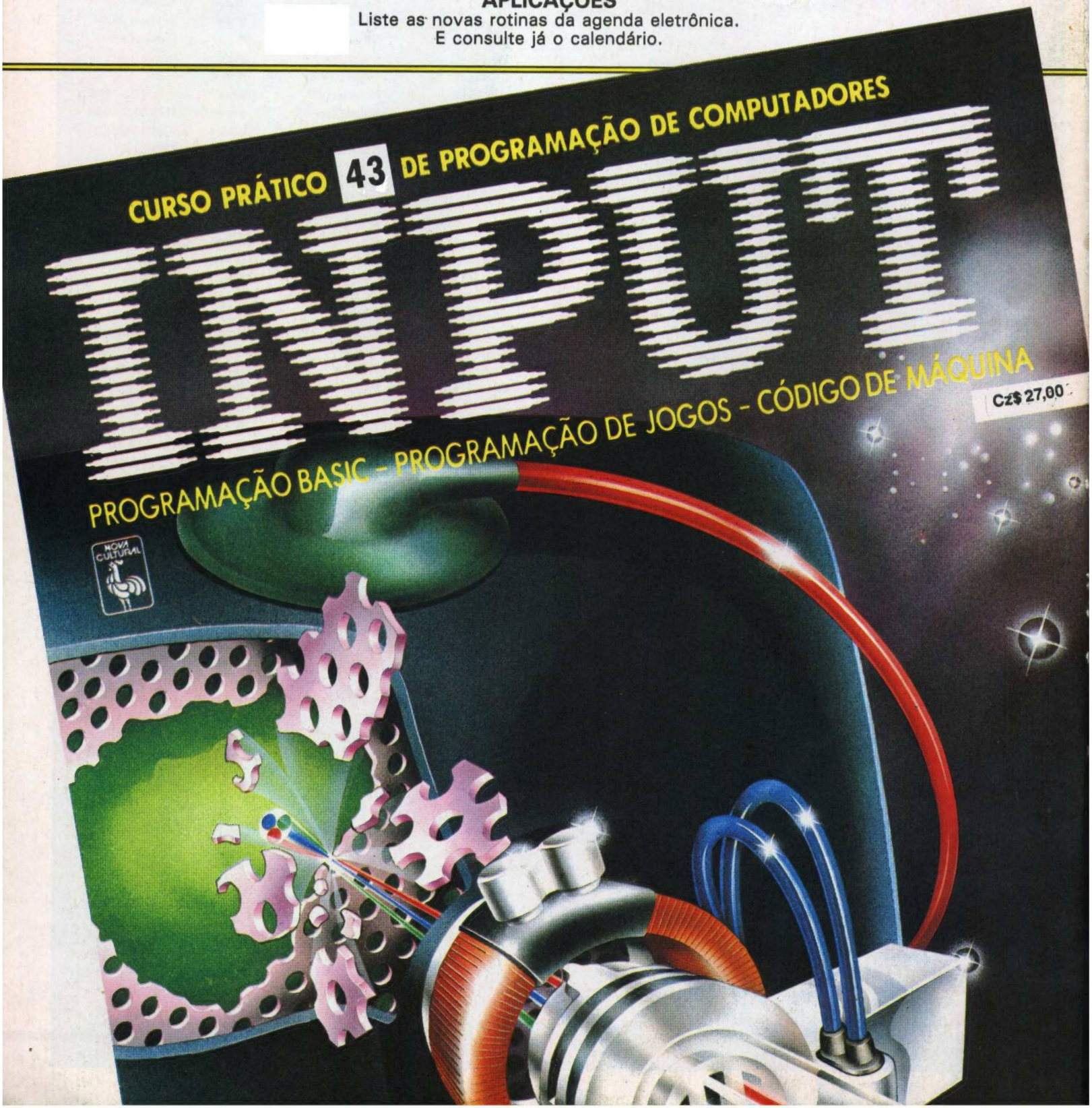
Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO DE JOGOS
Acione a manivela da máquina caça-níqueis de INPUT. Ela não vai deixá-lo de bolsos vazios.

PERIFÉRICOS
Monitores ou televisores? Para fazer uma boa escolha, conheça as características desses dois aparelhos.

PROGRAMAÇÃO BASIC
Com nosso programa "desenhista", transforme o vídeo de seu TK-2000 em uma tela de pintura.

APLICAÇÕES
Liste as novas rotinas da agenda eletrônica.
E consulte já o calendário.